ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΛΑΤΦΟΡΜΑΣ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ, ΜΕ ΣΕΒΑΣΜΟ ΣΤΙΣ ΑΠΑΙΤΗΣΕΙΣ ΤΗΣ ΙΔΙΩΤΙΚΟΤΗΤΑΣ ΤΩΝ ΧΡΗΣΤΩΝ

Του φοιτητή
Σωτήρη Τσεπελάκη
Αρ. Μητρώου: 134082

Επιβλέπων
Δρ. Περικλής Χατζημίσιος
Καθηγητής

Σεπτέμβριος 2022

Τίτλος Δ.Ε. ….
Κωδικός Δ.Ε. …
Ονοματεπώνυμο φοιτητή/τών ….
Ονοματεπώνυμο εισηγητή …
Ημερομηνία ανάληψης Δ.Ε. …
Ημερομηνία περάτωσης Δ.Ε. …

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Στους γονείς μου

Πρόλογος

Η παρούσα Πτυχιακή Εργασία εκπονήθηκε κατά την θερινή περίοδο του ακαδημαϊκού Έτους 2021 - 2022, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Διεθνούς Πανεπιστημίου της Ελλάδας (ΔΙΠΑΕ).

Η εργασία πραγματοποιήθηκε υπό την επίβλεψη του Καθηγητή κ. Περικλή Χατζημίσιου.

Αντικείμενο της εργασίας αποτελεί ο σχεδιασμός και η υλοποίηση ενός πρωτοτύπου για ασφαλή ανάλυση δεδομένων, με σεβασμό στην ιδιωτικότητα των χρηστών. Αυτό αποδεικνύεται χρήσιμο και αποτελεσματικό. Ο λόγος είναι ότι στην εποχή μας όπου είναι πλέον εκτεταμένη η χρήση και εφαρμογή της τεχνολογίας, των υπολογιστών και του διαδικτύου σε όλους σχεδόν τους τομείς, διαπιστώνεται συνεχώς η ανάγκη συγκέντρωσης πληροφοριών για την πραγματοποίηση διαφόρων μελετών που συντελούν στην εξέλιξή τους. Η ανάπτυξη των νέων τεχνολογιών και η ανάγκη ηλεκτρονικής οργάνωσης της έρευνας έχουν ως συνέπεια την αυξημένη ζήτηση προσωπικών πληροφοριών. Αποτέλεσμα αυτού είναι η αύξηση του κινδύνου μεταφοράς των δεδομένων αυτών. Σήμερα η άντληση γνώσεων χρησιμοποιείται από διάφορες εταιρίες, αφού στις μέρες μας παρατηρείται αυξημένη ζήτηση αυτού του είδους της τεχνολογίας. Θεώρησα λοιπόν ότι μπορώ να χρησιμοποιήσω τις γνώσεις που αποκόμισα κατά τη διάρκεια των σπουδών μου πάνω στον τομέα της επεξεργασίας των δεδομένων, να εξοικειωθώ με τις νέες τεχνολογίες ανάπτυξης και προγραμματισμού συστημάτων παροχής πληροφοριών και να αποκτήσω εμπειρία στην εφαρμογή αυτών των μεθόδων και εργαλείων.

Η σπουδαιότητα του θέματος έγκειται στο ότι η εξέλιξη των πραγμάτων ανέδειξε το πρόβλημα της προστασίας των προσωπικών δεδομένων ώστε ο χειρισμός τους να διενεργείται από τις επιβαλλόμενες συνθήκες ασφαλείας βάσει της νομοθεσίας της Ευρωπαϊκής Ένωσης η οποία δεσμεύει τα κράτη – μέλη να εφαρμόσουν ένα συγκεκριμένο νομικό πλαίσιο ώστε να προστατεύεται η ιδιωτική ζωή των ανθρώπων. Αποτελεί λοιπόν ένα επίκαιρο θέμα με το οποίο θεώρησα ότι αξίζει να ασχοληθώ, καθώς η επένδυση στην έρευνα βοηθά τους ανθρώπους να ζουν καλύτερα και παράλληλα να διασφαλίζονται τα δικαιώματά τους από τη διαχείριση συμφερόντων τρίτων.

Φυσικά και υπήρξαν δυσκολίες κατά την εκπόνηση της παρούσας πτυχιακής. Χρειάστηκε να μελετήσω μεταξύ άλλων, και εξειδικευμένα ιατρικά θέματα όπως οι κωδικοί και οι τιμές ιατρικών διαγνωστικών εξετάσεων κλπ. Επίσης δεν ήταν εύκολη η πρόσβαση για άντληση πληροφοριών από χώρους όπως οι βιβλιοθήκες εξαιτίας της υπάρχουσας κατάστασης του κορονοϊού.

Περίληψη

Η παρούσα πραγματεύεται την ανάγκη υιοθέτησης μίας πλατφόρμας για ασφαλή αποθήκευση και ανάλυση δεδομένων. Σκοπός της είναι η εύρεση και η έκθεση του τρόπου ανάλυσης των ευαίσθητων δεδομένων των χρηστών, με παράλληλη προστασία του απορρήτου τους.

Γενικά οι εταιρείες/οργανισμοί οφείλουν να διατηρούν την ιδιωτικότητα των χρηστών, ώστε να συμμορφώνονται με τον ΓΚΠΔ (*GDPR*), το οποίο ορίζει υψηλά πρότυπα προστασίας δεδομένων. Οι εταιρείες επιθυμούν να αναλύουν τα δεδομένα των χρηστών και να βεβαιώνουν συγχρόνως ότι η χρήση των δεδομένων συμμορφώνεται με την συγκατάθεση του χρήστη.

Οι υπάρχουσες προσεγγίσεις είναι οι εξής: Είτε οι εταιρείες δεν αναλύουν τα δεδομένα, είτε προσφέρουν εφαρμογές που τα αναλύουν τοπικά. Οι παραπάνω προσεγγίσεις συνεπάγονται απώλεια δυναμικής της ανάλυσης δεδομένων και εκμάθησης από ολόκληρα τα δεδομένα που συλλέγονται και ως εκ τούτου εμποδίζουν καινοτόμες έρευνες αλλά και υπηρεσίες. Σημαντικό παράδειγμα αποτελεί η αναγνώριση τάσεων σε πληθυσμούς χρηστών.

Στην παρούσα πτυχιακή εργασία θα καλυφθούν τα παραπάνω κενά, παραθέτοντας την προσέγγιση μιας κεντρικοποιημένης πλατφόρμας. Με αυτήν την προσέγγιση, οι εταιρίες από τη μία πλευρά μπορούν να επωφεληθούν από μια μεγάλη ομάδα ατόμων που είναι δυνητικά πρόθυμα να μοιραστούν τα δεδομένα τους για ευρύτερες εργασίες ανάλυσης, όπως αυτές που απαιτούνται π.χ. από ιατρική έρευνα. Οι χρήστες από την άλλη, επωφελούνται από μια πλατφόρμα διατήρησης του απορρήτου και της ασφάλειας των δεδομένων τους και μπορούν να συνεισφέρουν σε έργα ανάλυσης/έρευνας με ασφαλή τρόπο. Επιπλέον, οι αναλυτές (όπως ερευνητές) μπορούν να λάβουν μια ευρεία πηγή μικροδεδομένων, με την προϋπόθεση ότι οι χρήστες δίνουν την αντίστοιχη συναίνεση. Επιπροσθέτως θα αναζητηθούν και αναλυθούν τα δομικά-βασικά στοιχεία της αρχιτεκτονικής ενός τέτοιου συστήματος. Τέλος θα παρουσιαστεί η υλοποίηση ενός πρωτοτύπου και κατόπιν η αξιολόγησή του με περιπτώσεις χρήσης από πραγματικές εταιρίες.

ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ: ανάλυση δεδομένων, διατήρηση απορρήτου, προέλευση, διαχείριση συναίνεσης

DESIGN AND IMPLEMENTATION OF AN AUDITABLE PLATFORM THAT RESPECTS PRIVACY REQUIREMENTS OF USERS

SOTIRIOS TSEPELAKIS

Abstract

The thesis addresses the need to adopt a platform for secure data storage and analysis. Its purpose is to find and report on how to analyze users' sensitive data, whilst protecting their privacy. In general, companies/organizations must maintain the privacy of users in order to comply with the GDPR, which sets high standards of data protection. Companies want to analyze user data and at the same time ensure that the use of the data complies with the user's consent. Existing approaches are: Either companies do not analyze data, or they offer applications that analyze it locally. The above approaches imply a loss of dynamics of data analysis and learning from all the data collected and therefore hinder innovative research and services. An important example is the recognition of trends in user populations. In the present thesis the above gaps will be filled, citing the approach of a centralized platform. With this approach, companies on the one hand can benefit from a large group of people who are potentially willing to share their data for broader analysis tasks, such as those required e.g. from medical research. Users, on the other hand, benefit from a platform for maintaining the privacy and security of their data and can contribute to analytics / research projects in a secure way. In addition, analysts (such as researchers) can obtain a broad source of microdata, provided that users give their consent. In addition, the structural-basic elements of the architecture of such a system will be sought and analyzed. Finally, the implementation of a prototype will be presented and then its evaluation with use cases from real companies.

KEYWORDS: data analysis, privacy preservation, provenance, consent management

Ευχαριστίες

Η παρούσα εργασία εκπονήθηκε στο πλαίσιο της πτυχιακής εργασίας κατά το ακαδημαϊκό έτος 2021-2022.

Αντικείμενο της εργασίας αυτής αποτελεί η «Σχεδίαση και ανάπτυξη πλατφόρμας ανάλυσης δεδομένων, με σεβασμό στις απαιτήσεις της ιδιωτικότητας των χρηστών».

Στο σημείο αυτό της κορύφωσης των σπουδών μου οφείλω να αφιερώσω κάποιες σειρές στους ανθρώπους που συνέβαλαν στην εκπόνησή της εργασίας μου, ώστε να τους ευχαριστήσω για τη βοήθεια που μου προσέφεραν, γιατί αισθάνομαι ότι χωρίς τη βοήθειά τους δεν θα είχα εκπληρώσει τους στόχους μου.

Κατ' αρχάς, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας, Καθηγητή κ. Περικλή Χατζημίσιο για την καθοδήγηση, τις συμβουλές του και τον πολύτιμο χρόνο που διέθεσε όποτε τον χρειαζόμουν σε κάθε στάδιό της.

Επίσης τον συνεργάτη μου Dr. Tomasz Miksa, ο οποίος μου παρείχε την αμέριστη υποστήριξή και βοήθειά του κατά τη διάρκεια της εκπόνησής της.

Ακόμη, όλους τους αξιόλογους και αγαπημένους ανθρώπους που είχα δίπλα μου κατά τη διάρκεια των σπουδών μου και όλους όσους μου συμπαραστάθηκαν και με βοήθησαν με οποιοδήποτε τρόπο και όποτε τους χρειάστηκα.

Τέλος, ευχαριστώ την οικογένειά μου η οποία μου προσέφερε αφειδώλευτα την αγάπη της με υπομονή, στήριξη και ενθάρρυνση σε κάθε δύσκολη στιγμή μου, χωρίς την οποία θεωρώ ότι τίποτε από όσα έχω καταφέρει μέχρι σήμερα δεν θα είχα πραγματοποιήσει.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Κατάλογος Κώδικα

Συντομογραφίες

ΑΑΠ	Ασφαλής Αποθήκη Δεδομένων - Ασφαλές Αποθετήριο Δεδομένων
ΑΠΑΔ	Ασφαλές Περιβάλλον Ανάλυσης Δεδομένων
ΒΔ	Βάση Δεδομένων
ΓΚΠΔ	Γενικός Κανονισμός Προστασίας Δεδομένων
ΔΔΠΑ	Δημοσίευση Δεδομένων Προστασίας Απορρήτου
Δ.Ε.	Διπλωματική Εργασία
ΔΗΤ	Διεύθυνση Ηλεκτρονικού Ταχυδρομείου
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
ΔΠΔΙ	Διασύνδεση Πύλης Διακομιστή Ιστού
ΔΠΕ	Διεπαφή Προγραμματισμού Εφαρμογών (API)
ΔΧ	Διεπαφή Χρήστη
ΕΑΠ	Ενιαίο Αναγνωριστικό Πόρων
Ε.Ε.	Ευρωπαϊκή Ένωση
ΕΕΠ	Ενιαίος Εντοπιστής Πόρων
ΚΜΕ	Κεντρική Μονάδα Επεξεργασίας
ΜΜΕ	Μικρομεσαίες Εταιρίες
Π.Ε.	Πτυχιακή Εργασία
ΠΧ	Περίπτωση Χρήσης
ΣΧΑ	Σχεσιακός Χάρτης Αντικειμένων
ΥΡΠ	Υπηρεσία Ρυθμίσεων Πειραμάτων
AJAX	Asynchronous JavaScript and XML
ORM	Object Relational Mapper
PID	Persistent Identifier
RQ	Redis Queue
UUID	Universally Unique Identifier

Κεφάλαιο 1ο: Εισαγωγή

1.1 Γενικά

Από τον Μάιο του 2018 ο ΓΚΠΔ [1] ορίζει αυστηρά πρότυπα της προστασίας των δεδομένων και επιβάλλει σημαντικά πρόστιμα για την αποτυχία συμμόρφωσης με αυτά. Επιπλέον, οι παραβιάσεις ασφαλείας μπορούν να καταστρέψουν μόνιμα έναν οργανισμό, καθώς όταν χαθεί η εμπιστοσύνη και η φήμη, δύσκολα ανακτώνται.

Για αυτό λοιπόν το λόγο, δεν αρκεί μόνο για τους οργανισμούς να βεβαιωθούν ότι οι τεχνικές ανάλυσης δεδομένων των χρηστών εγγυώνται το απόρρητο [2]. Πρέπει ταυτοχρόνως να διασφαλίσουν ότι η χρήση αυτών των δεδομένων συμμορφώνεται και με την συγκατάθεση του χρήστη. Αυτό είναι ιδιαίτερα σημαντικό ώστε να παρακολουθείται ρητά η συναίνεση για τα ευαίσθητα δεδομένα, όπως ορίζει και ο ΓΚΠΔ. Επομένως χωρίς ισχυρό υπόβαθρο στην ασφάλεια πληροφοριακών συστημάτων και χωρίς μεγάλους προϋπολογισμούς, η παροχή μιας ασφαλούς πλατφόρμας για αποθήκευση και ανάλυση δεδομένων είναι συχνά πέρα από τις δυνατότητες των μικρομεσαίων οργανισμών.

Κατά συνέπεια, είτε οι εταιρείες δεν αναλύουν τα δεδομένα, είτε προσφέρουν εφαρμογές που τα αναλύουν τοπικά. Οι παραπάνω προσεγγίσεις συνεπάγονται σε απώλεια δυναμικής της ανάλυσης δεδομένων και εκμάθησης από ολόκληρα τα δεδομένα που συλλέγονται. Ως εκ τούτου εμποδίζουν καινοτόμες έρευνες αλλά και υπηρεσίες, όπως την αναγνώριση τάσεων σε πληθυσμούς χρηστών.

Δεν υπάρχει λοιπόν κάποια κεντρικοποιημένη πλατφόρμα, η οποία να παρέχει τη δυνατότητα ανάλυσης προσωπικών δεδομένων, χωρίς να αποκαλύπτεται η ταυτότητα του χρήστη στους αναλυτές. Συνεπώς το κύριο ερώτημα μπορεί να διατυπωθεί ως εξής: "Πώς μπορεί να γίνει λειτουργικό ένα ελεγχόμενο σύστημα ανάλυσης δεδομένων, το οποίο θα προστατεύει την ιδιωτικότητα;"

Στην παρούσα λοιπόν θα καλυφθούν τα παραπάνω κενά, παραθέτοντας την προσέγγιση μιας κεντρικοποιημένης πλατφόρμας. Κύρια χαρακτηριστικά της πλατφόρμας αυτής θα είναι η παροχή α) ασφαλούς αποθήκευσης για τα ευαίσθητα δεδομένα των χρηστών με ρητή συγκατάθεση και β) ενός αξιόπιστου περιβάλλοντος ανάλυσης αυτών με σεβασμό στην ιδιωτικότητα των χρηστών. Η καινοτομία που εισάγεται είναι ότι οι οργανισμοί δεν έχουν άμεση πρόσβαση στα προσωπικά δεδομένα παρά μόνο σε συγκεντρωτική *(aggregated)* ή ανωνυμοποιημένη μορφή. Με αυτήν την προσέγγιση, οι εταιρίες από τη μία πλευρά μπορούν να επωφεληθούν από μια μεγάλη ομάδα ατόμων που είναι δυνητικά πρόθυμα να μοιραστούν τα δεδομένα τους για ευρύτερες εργασίες ανάλυσης, όπως αυτές που απαιτούνται π.χ. από ιατρική έρευνα. Οι χρήστες από την άλλη, επωφελούνται από μια πλατφόρμα διατήρησης του απορρήτου και της ασφάλειας των δεδομένων τους και μπορούν να συνεισφέρουν σε έργα ανάλυσης/έρευνας με ασφαλή τρόπο. Επιπλέον, οι αναλυτές/ερευνητές μπορούν να λάβουν μια ευρεία πηγή μικροδεδομένων, με την προϋπόθεση ότι οι χρήστες δίνουν την αντίστοιχη συναίνεση.

1.2 Προσεγγίσεις

Ορισμένες προσεγγίσεις, επιτρέπουν την ανάλυση δεδομένων που διανέμονται μεταξύ πολλαπλών πηγών, υπολογίζοντας ένα κοινό αποτέλεσμα, χωρίς να χρειάζεται να συγκεντρωθούν οι εισροές (δεδομένα). Ωστόσο, οι περισσότερες από αυτές τις προσεγγίσεις διερευνώνται καλά μόνο όταν τα δεδομένα είναι οριζόντια διαχωρισμένα *(horizontally partitioned)*. Στο σενάριό μας, έχουμε γενικά μια

ρύθμιση με κατακόρυφο διαχωρισμό *(vertically partitioned)*, δηλαδή έχουμε αρχεία δεδομένων που περιγράφουν διαφορετικές πτυχές του ίδιου ατόμου(ων), κάτι που εξακολουθεί να δημιουργεί ορισμένες προκλήσεις. Ως εκ τούτου, εστιάζουμε στην κεντρική ρύθμιση που προτείνεται στην εργασία μας. Για το σκοπό αυτό, το κύριο ερευνητικό πρόβλημα που συζητείται σε αυτό την εργασία μπορεί να διατυπωθεί ως εξής: *Πώς μπορούμε να δημιουργήσουμε ένα σύστημα ανάλυσης δεδομένων με δυνατότητα ελέγχου και διατήρησης της ιδιωτικότητας των χρηστών της;* Επιπλέον, χωρίσαμε το πρόβλημα σε μια σειρά από ερευνητικά ερωτήματα ως εξής:

- *Ποια είναι τα βασικά χαρακτηριστικά ενός συστήματος ανάλυσης δεδομένων με σεβασμό στην ιδιωτικότητα των χρηστών της;*
- *Ποια είναι τα βασικά στοιχεία μιας αρχιτεκτονικής ενός συστήματος ανάλυσης δεδομένων με σεβασμό στην ιδιωτικότητα των χρηστών της;*
- *Πώς μπορούν να χρησιμοποιηθούν τεχνολογίες για να υλοποιηθεί η αρχιτεκτονική ενός συστήματος ανάλυσης δεδομένων με σεβασμό στην ιδιωτικότητα των χρηστών της;*

1.3 Επισκόπηση σχετικής έρευνας

Πρόσφατα, η έρευνα για την αντιμετώπιση προκλήσεων που σχετίζονται με τη δυνατότητα ελέγχου και το απόρρητο των χρηστών έχει ενταθεί. Η ανάλυση δεδομένων με σεβασμό στην ιδιωτικότητα είναι ένας ενεργός ερευνητικός τομέας στην ερευνητική κοινότητα. Από τη μία πλευρά, η εστίασή του είναι στον έλεγχο του χειρισμού των προσωπικών δεδομένων για την αποφυγή προβλημάτων απορρήτου. Από την άλλη πλευρά, η μεγιστοποίηση της χρησιμότητας των δεδομένων είναι μια απαίτηση για μια αποτελεσματική μέθοδο διατήρησης της ιδιωτικότητας. Διάφορες προσεγγίσεις που έχουν προταθεί για την ανάλυση δεδομένων με διατήρηση της ιδιωτικότητας μπορούν να κατηγοριοποιηθούν σύμφωνα με τον κύκλο ζωής των δεδομένων στον οποίο εφαρμόζονται. Για παράδειγμα: *Δημοσίευση Δεδομένων με Σεβασμό στην Ιδιωτικότητα* (*Privacy-Preserving Data Publishing - PPDP*) και *Αποτελέσματα Εξόρυξης Δεδομένων με Σεβασμό στην Ιδιωτικότητα* (*Privacy-Preserving Data Mining Outputs - PPDMO*) [3]. Στην εργασία βασιζόμαστε στις μεθόδους *PPDP* και *PPDMO*. Το *PPDP* βασίζεται στην ανωνυμοποίηση των εγγραφών δεδομένων πριν από τη δημοσίευση. Η πιο σημαντική μέθοδος είναι η k-ανωνυμοποίηση (*k-anonymity*). Οι προσεγγίσεις για το *PPDMO* περιλαμβάνουν τον *Έλεγχο Αποτελεσμάτων Ερωτήματος (Query Inference Control)*, όπου είτε τα αρχικά δεδομένα, είτε η έξοδος του ερωτήματος υπόκεινται σε αλλαγές (θόρυβο) και τον *Έλεγχο Ερωτήματος* (*Query Auditing)* όπου ορισμένα ερωτήματα απορρίπτονται λόγω πιθανής παραβίασης του απορρήτου. Το DataSHIELD [4] είναι ένα παράδειγμα τέτοιου συστήματος, το οποίο εφαρμόζει και τις δύο μεταγενέστερες τεχνικές σε έναν αριθμό συναρτήσεων εξόρυξης δεδομένων (περισσότερες λεπτομέρειες στα κεφάλαια 2 και 4). Παρά την έρευνα σε μεμονωμένα θέματα, δεν υπάρχει ολοκληρωμένη αρχιτεκτονική για ελεγχόμενη ανάλυση δεδομένων διατήρησης της ιδιωτικότητας. Η έλλειψη μιας τέτοιας προσέγγισης μπορεί να αποδοθεί εν μέρει, στους αντικρουόμενους στόχους της συλλογής, όσο το δυνατόν περισσότερων δεδομένων για δυνατότητα ελέγχου, ενώ στοχεύει στη μειωμένη συλλογή δεδομένων για την ελαχιστοποίηση των επιπτώσεων μιας παραβίασης δεδομένων. Περισσότερες λεπτομέρειες για ιδέες, εργαλεία και συστήματα που συνάδουν με την παρούσα, παραθέτονται στο κεφάλαιο 2.

1.4 Λύση

Για να καλυφθεί αυτό το κενό, προτείνουμε μια νέα προσέγγιση με μια αρχιτεκτονική για ελεγχόμενη ανάλυση δεδομένων που διατηρεί την ιδιωτικότητα, η οποία (i) παρέχει ασφαλή αποθήκευση για ευαίσθητα δεδομένα των χρηστών με ρητή συγκατάθεση και (ii) παρέχει ένα αξιόπιστο περιβάλλον ανάλυσης για την εκτέλεση διεργασιών ανάλυσης δεδομένων, με διατήρηση της ιδιωτικότητας των

χρηστών. Η προσέγγισή μας είναι καινοτόμα, γιατί οι οργανισμοί δεν έχουν άμεση πρόσβαση σε μεμονωμένα δεδομένα, αλλά έχουν πρόσβαση σε αυτά μόνο σε συγκεντρωτική (*aggregated*) ή ανωνυμοποιημένη μορφή. Οι οργανισμοί από τη μία πλευρά μπορούν να επωφεληθούν από μια μεγάλη ομάδα ατόμων που είναι δυνητικά πρόθυμοι να μοιραστούν τα δεδομένα τους για ευρύτερες εργασίες ανάλυσης, όπως αυτές που απαιτούνται για παράδειγμα από την ιατρική έρευνα. Οι χρήστες από την άλλη πλευρά, επωφελούνται από μια πλατφόρμα που διατηρεί το απόρρητο και παρέχει ασφάλεια για τα δεδομένα τους και μπορούν να συνεισφέρουν σε αναλυτικά/έρευνα έργα με ασφαλή τρόπο. Τέλος, οι αναλυτές (όπως οι ερευνητές) μπορούν να λάβουν μια λεπτομερή πηγή μικροδεδομένων, εάν οι κάτοχοι των δεδομένων δώσουν την αντίστοιχη συγκατάθεση.

Η συμβολή της παρούσας είναι τριπλή:

- Προτείνουμε μια αρχιτεκτονική με ασφαλή αποθήκευση αλλά και ανάλυση δεδομένων με σεβασμό στην ιδιωτικότητα.
- Παρουσιάζουμε την υλοποίηση ενός πρωτοτύπου βασιζόμενο σε αυτήν την αρχιτεκτονική, προσαρμόζοντας διάφορες τεχνολογίες συμπεριλαμβανομένων υπηρεσιών ιστού, αναπαράστασης μεταδεδομένων και δεδομένων προέλευσης.
- Επιδεικνύουμε τη σκοπιμότητα της προσέγγισής μας με πραγματικές περιπτώσεις χρήσης στον ιατρικό τομέα.

1.5 Επισκόπηση της Π.Ε.

Η υπόλοιπη εργασία δομείται ως ακολούθως:

- Στο 2ο κεφάλαιο παρουσιάζεται η σχετική έρευνα, με διάφορες ιδέες και τεχνολογίες. Το κεφάλαιο αυτό περιέχει κομμάτια σχετικά με την προσέγγιση της παρούσας αλλά και κομμάτια τα οποία μπορούν να χρησιμοποιηθούν για τη βελτίωση της.

- Στο 3ο κεφάλαιο περιγράφονται οι απαιτήσεις και η αρχιτεκτονική του συστήματος.

- Στο 4ο κεφάλαιο παρουσιάζεται η υλοποίηση του πρωτοτύπου.

- Στο 5ο κεφάλαιο αναλύεται η αξιολόγηση του πρωτοτύπου με διάφορες περιπτώσεις χρήσης.

- Με το 6ο κεφάλαιο ολοκληρώνεται η εργασία, όπου παρατίθεται ο επίλογος και τα συμπεράσματα που προκύπτουν.

Κεφάλαιο 2ο: Σχετική Έρευνα

2.1 Εισαγωγή

Αυτό το κεφάλαιο εξετάζει τη σχετική εργασία από τρεις οπτικές γωνίες: ανάλυση δεδομένων με σεβασμό στην ιδιωτικότητα (Ενότητα 2.2), δυνατότητα ελέγχου (Ενότητα 2.3) και διαχείριση συναίνεσης (Ενότητα 2.4). Οι επόμενες ενότητες παραθέτουν αναφορές σε ιδέες, project, άρθρα και υλοποίησεις που είναι σχετικές με την εργασία μας. Μερικές αναφορές είναι εκτός βελησνεκούς της τρέχουσας εργασίας, παρ' όλα αυτά είναι σχετικές με ολόκληρο το σύστημα και μπορούν πιθανώς να χρησιμοποιηθούν για τη βελτίωσή του.

2.2 Ανάλυση δεδομένων με σεβασμό στην ιδιωτικότητα

Για την επίτευξη ανάλυσης δεδομένων με διατήρηση της ιδιωτικότητας, συχνά εξετάζονται δύο συμπληρωματικές προσεγγίσεις: δημοσίευση δεδομένων με σεβασμό στην ιδιωτικότητα (*Privacy Preserving Data Publishing-PPDP*) [5] και υπολογισμοί με σεβασμό στην ιδιωτικότητα (*Privacy Preserving Computation)*, όπως ασφαλής υπολογισμός πολλαπλών μερών (*Secure-Multiparty Computation-SMPC)* ή ομομορφική κρυπτογράφηση (*homomorphic encryption*).

Όσον αφορά το PPDP, οι πρώιμες προσεγγίσεις περιλαμβάνουν, π.χ. k-anonymity [6], όπου σε αυτήν την προσέγγιση αντιμετωπίζουν το πρόβλημα της προστασίας του απορρήτου στην απελευθέρωση πληροφοριών και παρουσιάζουν μια προσέγγιση για την αποκάλυψη μικροδεδομένων, έτσι ώστε οι ταυτότητες των ερωτηθέντων να μην μπορούν να αναγνωριστούν. Η απαίτηση ανωνυμοποίησης εκφράζεται με τον καθορισμό ενός αριθμού k που δηλώνει την απαιτούμενη προστασία. Η επιβολή k-ανωνυμοποίησης σημαίνει διασφάλιση ότι ο παραλήπτης πληροφοριών δεν θα είναι σε θέση, ακόμη και όταν συνδέει πληροφορίες με εξωτερικά δεδομένα, να συσχετίσει κάθε πλειάδα που κυκλοφόρησε με λιγότερα από k άτομα. Εκτός αυτού, επεξηγείται πώς μπορεί να μεταφραστεί η απαίτηση k-ανωνυμοποίησης, μέσω της έννοιας των quasi-αναγνωριστικών. Ακόμη επεξηγείται πώς η k-ανωνυμοποίηση μπορεί να επιβληθεί χρησιμοποιώντας τεχνικές γενίκευσης (*generalization*) (την οποία χρησιμοποιούμε στην υλοποίηση του πρωτοτύπου) και καταστολής (*suppression*). Οι συγγραφείς επισημαίνουν ότι ένα πρωτότυπο του συστήματος τους είναι υπό υλοποίηση. Το κύριο κομμάτι της προαναφερθείσας εργασίας είναι ο μετασχηματισμός των δεδομένων, τέτοιος ώστε οι μεμονωμένες εγγραφές να μην μπορούν να διακριθούν από τις άλλες k-1 εγγραφές, συνεπώς ο επαναπροσδιορισμός γίνεται δύσκολος. Αρκετές υλοποιήσεις εκτελούν αυτούς τους μετασχηματισμούς, π.χ., ARX [7].

Το διαφορικό απόρρητο (*differential privacy*) [8] στοχεύει στη μεγιστοποίηση της ακρίβειας των απαντήσεων σε ερωτήματα στις βάσεις δεδομένων, ελαχιστοποιώντας παράλληλα την πιθανότητα να είναι δυνατή η αναγνώριση των εγγραφών που χρησιμοποιούνται για την απάντηση τους. Εάν ένα συγκεκριμένο αντικείμενο περιλαμβάνεται στο σύνολο δεδομένων, δεν θα πρέπει να "αυξήσει" το απόρρητο του ατόμου που εκπροσωπείται. Τα συστήματα που εφαρμόζουν διαφορικό απόρρητο περιλαμβάνουν:

(i) το GUPT [9], το οποίο χρησιμοποιεί ένα μοντέλο ευαισθησίας δεδομένων, που πραγματοποιεί "*degradation*" στο απόρρητο των δεδομένων με την πάροδο του χρόνου (δηλαδή κάνει τα δεδομένα λιγότερο ευαίσθητα). Αυτό επιτρέπει την αποτελεσματική κατανομή διαφορετικών επιπέδων απορρήτου για διαφορετικές εφαρμογές χρηστών, ενώ παράλληλα εγγυάται ένα συνολικό σταθερό επίπεδο απορρήτου και μεγιστοποιεί τη χρησιμότητα κάθε εφαρμογής. Το GUPT εισάγει επίσης τεχνικές που βελτιώνουν την ακρίβεια των αποτελεσμάτων, επιτυγχάνοντας το ίδιο επίπεδο

ιδιωτικότητας. Αυτές οι προσεγγίσεις του επιτρέπουν να εκτελεί εύκολα μια μεγάλη ποικιλία προγραμμάτων ανάλυσης δεδομένων, παρέχοντας ταυτόχρονα χρησιμότητα και ιδιωτικότητα.

(ii) το PINQ [10], το οποίο είναι μια επεκτάσιμη πλατφόρμα ανάλυσης δεδομένων που έχει σχεδιαστεί για να παρέχει άνευ όρων εγγυήσεις απορρήτου για τα αρχεία των υποκείμενων συνόλων δεδομένων. Το PINQ παρέχει στους αναλυτές πρόσβαση σε εγγραφές μέσω μιας δηλωτικής γλώσσας τύπου SQL (LINQ) ανάμεσα σε κατά τα άλλα, arbitrary κώδικα C#. Ταυτόχρονα, ο σχεδιασμός της γλώσσας ανάλυσης του PINQ και η προσεκτική εφαρμογή της παρέχουν επίσημες εγγυήσεις διαφορικού απορρήτου για οποιαδήποτε χρήση της πλατφόρμας. Οι εγγυήσεις του PINQ δεν απαιτούν καμία εμπιστοσύνη στην τεχνογνωσία ή την επιμέλεια των αναλυτών, διευρύνοντας το πεδίο σχεδιασμού και ανάπτυξης αναλύσεων δεδομένων που διατηρούν το απόρρητο, ειδικά από μη ειδικούς σε θέματα απορρήτου.

(iii) το Airavat [11], ένα σύστημα που βασίζεται στο MapReduce, το οποίο παρέχει ισχυρές εγγυήσεις ασφάλειας και απορρήτου για κατανεμημένους υπολογισμούς σε ευαίσθητα δεδομένα. Με αυτό το σύστημα, οι πάροχοι δεδομένων ελέγχουν την πολιτική ασφαλείας για τα ευαίσθητα δεδομένα τους, συμπεριλαμβανομένου ενός μαθηματικού ορίου σχετικά με πιθανές παραβιάσεις του απορρήτου. Οι χρήστες, χωρίς εξειδίκευση σε θέματα ασφάλειας, μπορούν να εκτελούν υπολογισμούς στα δεδομένα, αλλά το Airavat περιορίζει τους υπολογισμούς, αποτρέποντας τη διαρροή πληροφοριών πέρα από την πολιτική του παρόχου δεδομένων.

Πρόσφατα, η σύνθεση δεδομένων εμφανίζεται ως εναλλακτική μέθοδος [12]. Η σύνθεση δεδομένων δημιουργεί ένα τεχνητό σύνολο δεδομένων που ωστόσο εξακολουθεί να διατηρεί τις καθολικές ιδιότητες των αρχικών δεδομένων. Μια προσέγγιση είναι, π.χ. το Synthetic Data Vault (SDV) [13]. Το SDV δημιουργεί ένα μοντέλο των δεδομένων με βάση τις εκτιμήσεις για τις κατανομές κάθε στήλης. Προκειμένου να διατηρηθεί η συσχέτιση μεταξύ των χαρακτηριστικών, ο "συνθέτης" (*synthesizer*) εφαρμόζει μια πολυμεταβλητή έκδοση του ζεύγους Gauss (*Gaussian Copula*) και, στη συνέχεια, υπολογίζει τον πίνακα συνδιακύμανσης. Αυτό το μοντέλο χρησιμοποιείται στη συνέχεια για τη δημιουργία νέων, συνθετικών δειγμάτων. Η σύνθεση δεδομένων χρησιμοποιείται και στην υλοποίηση του πρωτοτύπου μας.

Ένα αρκετά παρόμοιο project με την παρούσα είναι το OSSDIP [14]. Σε αυτό παρουσιάζεται μια αρχιτεκτονική και διαδικασίες για τη λειτουργία μιας ασφαλούς υποδομής δεδομένων, που υποστηρίζει την ασφαλή επίσκεψη δεδομένων. Οι κάτοχοι δεδομένων μπορούν να κάνουν τα δεδομένα τους προσβάσιμα σε αναξιόπιστους επισκέπτες (ειδικούς, αναλυτές) για συγκεκριμένες εργασίες, διατηρώντας παράλληλα τον πλήρη έλεγχο του τρόπου χρήσης των δεδομένων τους και αποτρέποντας τη διαρροή δεδομένων. Η ιδέα βασίζεται στην παροχή πρόσβασης στα υποσύνολα (δυνητικά δακτυλικών αποτυπωμάτων ή/και ανωνυμοποιημένων) δεδομένων, που απαιτούνται για μια συγκεκριμένη δραστηριότητα, μέσω απομονωμένων *εικονικών μηχανών* (*virtual machines*), οι οποίες παρακολουθούνται και είναι προσβάσιμες μόνο μέσω απομακρυσμένης πρόσβασης στην επιφάνεια εργασίας. Επιπλέον, παρουσιάζεται μια υλοποίηση αυτής της υποδομής που βασίζεται εξ ολοκλήρου σε καλά μελετημένα στοιχεία ανοιχτού κώδικα, της οποίας η εγκατάσταση, η διαμόρφωση και οι βασικές λειτουργικές διαδικασίες έχουν αυτοματοποιηθεί σε τέτοιο βαθμό ώστε να μπορούν να αναπτυχθούν εύκολα μέσα σε σύντομο χρονικό διάστημα σε οποιαδήποτε υποσχόμενη υποδομή ή ιδιωτικό cloud.

Ενώ αυτές οι λύσεις παρέχουν δομικά στοιχεία ανάλυσης δεδομένων που διατηρούν το απόρρητο, δεν παρέχουν ένα πλήρες σύστημα που περιλαμβάνει ταυτόχρονα συλλογή δεδομένων, διαχείριση συναίνεσης, μετασχηματισμό δεδομένων και την ανάλυση δεδομένων διατήρησης της ιδιωτικότητας.

Μια πρόσφατη αναφορά για τα Έμπιστα Ερευνητικά Περιβάλλοντα [15] επεξεργάζεται τις απαιτήσεις για τέτοια περιβάλλοντα στον τομέα της υγείας.

Ένα σύστημα που χρησιμοποιεί blockchain για την ανίχνευση της χρήσης ενοποιημένων πηγών δεδομένων σε μια διαδικασία ανάλυσης παρουσιάζεται στο [16], παρέχοντας επίσης έναν μηχανισμό ελέγχου για συναίνεση.

Το project DEXHELPP [17] εστιάζει σε ρυθμίσεις όπως ασφάλεια, π.χ., επιτρέποντας στον χρήστη μια σχετικά απεριόριστη πρόσβαση σε δεδομένα μέσα σε ένα συγκεκριμένο απομακρυσμένο υπολογιστικό περιβάλλον, αλλά δεν αντιμετωπίζει ζητήματα ενοποίησης δεδομένων ή διαχείρισης συναίνεσης.

Το DataSHIELD [4] είναι ένα σύστημα που επιτρέπει την ανάλυση δεδομένων από ενοποιημένες πηγές (*federated sources*), προστατεύοντας παράλληλα το απόρρητο με το να περιορίζει τον αναλυτή να εκτελεί μόνο συγκεντρωτικά ερωτήματα. Δεν εξετάζει πτυχές της διαχείρισης δεδομένων και συναίνεσης. Στο πρωτότυπο μας, χρησιμοποιούμε το DataSHIELD για να παρέχουμε τη λειτουργικότητα του ΑΠΑΔ.

Ένα παρόμοιο σύστημα είναι το ViPAR [18], μια ασφαλής πλατφόρμα ανάλυσης για ενοποιημένα δεδομένα (*federated data*). Το σύστημα προσφέρει έναν κεντρικό διακομιστή, όπου τα δεδομένα συγκεντρώνονται εικονικά μέσω του πρωτοκόλλου Secure Shell από απομακρυσμένα ερευνητικά σύνολα δεδομένων, που φιλοξενούνται από ερευνητικά ιδρύματα. Τα δεδομένα μπορούν στη συνέχεια να χρησιμοποιηθούν για ανάλυση μέσω μιας ασφαλούς πύλης ανάλυσης και διαγράφονται μόλις ολοκληρωθεί η ανάλυση. Το κεντρικό στοιχείο συγκέντρωσης είναι η κύρια διαφορά μεταξύ DataSHIELD και ViPAR. Το DataSHIELD εφαρμόζει την ανάλυση σε καθεμία από τις απομακρυσμένες πηγές χωριστά και στη συνέχεια συνδυάζει τα αποτελέσματα της ανάλυσης, ενώ το ViPAR συνδυάζει τα ίδια τα δεδομένα.

Το project BioSHaRE [19] (Biobank Standardization and Harmonization for Research Excellence in the European Union) είναι ένα συλλογικό ευρωπαϊκό έργο που επικεντρώνεται στην ανάπτυξη εργαλείων για την εναρμόνιση δεδομένων, την ενοποίηση βάσεων δεδομένων και τη *Συνεργατική Προσέγγιση Μάθησης* (*Federated Learning*) για να επιτρέψει μελέτες μεγάλης κλίμακας σε πολλά ιδρύματα. Το project αυτό περιλαμβάνει το DataSHIELD [4] ως εργαλείο ανάλυσης που προστατεύει το απόρρητο. Άλλα συστήματα που χρησιμοποιούνται στο project είναι το DataHSaPER για συνένωση και εναρμόνιση βάσεων δεδομένων και το OBiBa ως εργαλεία τεχνολογίας πληροφοριών.

Εάν τα δεδομένα διανέμονται μεταξύ πολλών πηγών, ο υπολογισμός ενός κοινού αποτελέσματος χωρίς να χρειάζεται να μοιραζόμαστε τα δεδομένα μπορεί να λύσει τις απαιτήσεις ιδιωτικότητας. Για παράδειγμα, ο ασφαλής υπολογισμός πολλαπλών μερών παρέχει ένα κρυπτογραφικό πρωτόκολλο για τον υπολογισμό της εξόδου χωρίς την ανάγκη τρίτου μέρους. Άλλες πιο ελαφριές προσεγγίσεις, όπως η *Συνεργατική Προσέγγιση Μάθησης* (*Federated Learning*) [20] έχουν προταθεί ως πιθανή λύση. Η *Συνεργατική Προσέγγιση Μάθησης* στοχεύει γενικά στην εκμάθηση ενός παγκόσμιου μοντέλου συγκεντρώνοντας τοπικά εκπαιδευμένα μοντέλα. Με αυτόν τον τρόπο, τα ανεπεξέργαστα δεδομένα δεν χρειάζεται να φύγουν από τη συσκευή όπου είναι αποθηκευμένα.

Ωστόσο, οι περισσότερες προσεγγίσεις, όπως η *Συνεργατική Προσέγγιση Μάθησης* (*Federated Learning*), διερευνώνται ως επί το πλείστον όταν τα δεδομένα είναι *οριζόντια διαχωρισμένα* (*horizontally partitioned*), δηλαδή όταν υπάρχουν πολλοί ιστότοποι που διαθέτουν δεδομένα που περιγράφονται από τα ίδια χαρακτηριστικά, αλλά περιέχουν διαφορετικές μεμονωμένες εγγραφές. Στο σενάριό μας, όταν συνδυάζουμε δεδομένα από πολλαπλές προελεύσεις, έχουμε γενικά μια ρύθμιση *κάθετου διαχωρισμού* (*vertical partitioning*), δηλαδή έχουμε αρχεία δεδομένων που περιγράφουν

διαφορετικές πτυχές του ίδιου ατόμου(ων). Μια τέτοια ενοποίηση και σύνδεση πολλαπλών πηγών δεδομένων, λαμβάνοντας υπόψη και τον έλεγχο συναίνεσης, εξακολουθεί να δημιουργεί προκλήσεις, όπως η διατήρηση μυστικότητας του υποκειμένου (*subject*) [21]. Επιπλέον, η δυνατότητα ελέγχου είναι δυσκολότερη να επιτευχθεί στο ενοποιημένο περιβάλλον (*federated environment*). Ως εκ τούτου, εστιάζουμε στην κεντρική ρύθμιση που προτείνεται στην παρούσα.

2.3 Δυνατότητα Ελέγχου

Παραδοσιακά, τα αρχεία καταγραφής συστημάτων συλλέγονται και αναλύονται για τη διεξαγωγή ελέγχων συστημάτων και, ως εκ τούτου, για να γίνουν τα υποκείμενα συστήματα πιο "διαφανή". Ωστόσο, η ανάλυση τέτοιων αρχείων καταγραφής είναι επίπονη και κουραστική. Επιπλέον, το δικαίωμα της εξήγησης, όπως υποστηρίζεται από τον ΓΚΠΔ της Ε.Ε. (*GDPR*) [1], απαιτεί καταλληλότερες μεθόδους παροχής πλαισίου (*context*) και αιτιολόγησης. Ως εκ τούτου, οι σημασιολογικές τεχνολογίες φαίνεται να είναι μια πολλά υποσχόμενη μέθοδος για τη σύλληψη και τη διαχείριση δεδομένων με δομημένο τρόπο και για την παροχή υποστήριξης στην αντιμετώπιση ερωτημάτων ελέγχου. Τονίζουμε πως στο σύστημά μας δεν θα υιοθετήσουμε σημασιολογικές τεχνολογίες ή οντολογίες, αλλά πραγματοποιούμε αναφορές σε αυτές, καθώς μπορούν να θεωρηθούν ως μελλοντική εργασία.

Διάφορα framework και λύσεις για την καταγραφή ροών (*workflows*), όπως το Taverna [22] ή μοντέλα δεδομένων για τη συλλογή δεδομένων προέλευσης όπως το PROV-DM [23] επιτρέπουν βελτιωμένη ανάλυση και διαχείριση.

Τα δεδομένα προέλευσης μπορούν επίσης να υποστηρίξουν την αναπαραγωγιμότητα (*reproducibility*) των αποτελεσμάτων της έρευνας, καταγράφοντας το πλαίσιο του πειράματος, για παράδειγμα χρησιμοποιώντας οντολογίες [24-26].

Λύσεις για συγκεκριμένες γλώσσες προγραμματισμού, όπως η noworkflow [27] για script python ή rdt για script R, έχουν αναπτυχθεί και προσφέρουν πλουσιότερη υποστήριξη, όπως οπτικοποίηση δεδομένων (*data visualization*) [28].

Το Prov-O χρησιμεύει ως θεμελιώδες δομικό στοιχείο για τη συλλογή δεδομένων που σχετίζονται με την προέλευση. Ωστόσο, ο γενικός χαρακτήρας του απαιτεί επέκταση και προσαρμογή σε συγκεκριμένα περιβάλλοντα για να μπορέσει να επωφεληθεί από τις συμφραζόμενες πληροφορίες (*contextual information*).

Το ProvStore, που αναπτύχθηκε ως το πρώτο δημόσιο αποθετήριο εγγράφων προέλευσης, δείχνει την ανάγκη για μια τέτοια βελτιωμένη διαχείριση δεδομένων [29].

Οι περιπτώσεις χρήσης που σχετίζονται με τον έλεγχο και την προέλευση περιλαμβάνουν τη δυνατότητα απόδειξης της συμμόρφωσης με τη νομοθετική ρύθμιση (π.χ. GDPR) όπως προτείνεται από το [30] για την οπτικοποίηση εξαρτήσεων ή δυνατοτήτων εντοπισμού σφαλμάτων [31].

Ωστόσο, μέχρι στιγμής, τα περισσότερα εργαλεία και προσεγγίσεις είναι πολύ συγκεκριμένα για ορισμένα περιβάλλοντα ή ρυθμίσεις χωρίς να ενσωματώνουν διαδρομές προέλευσης.

2.4 Διαχείριση συναίνεσης

Ο ΓΚΠΔ (GDPR) της Ε.Ε. [1] ορίζει ένα σύνολο υποχρεώσεων για τους υπεύθυνους επεξεργασίας προσωπικών δεδομένων. Μεταξύ άλλων απαιτήσεων, ο ΓΚΠΔ απαιτεί από τους υπευθύνους επεξεργασίας δεδομένων να λαμβάνουν ρητή συγκατάθεση για την επεξεργασία προσωπικών δεδομένων από τα υποκείμενα των δεδομένων. Επιπλέον, οι οργανισμοί πρέπει να είναι ρητοί σχετικά

με την επεξεργασία των προσωπικών τους δεδομένων και να αποδεικνύουν ότι τα συστήματά τους συμμορφώνονται με τους περιορισμούς χρήσης που καθορίζονται από τα υποκείμενα των δεδομένων (*data subjects*).

Η παραδοσιακή αναπαράσταση της συναίνεσης του χρήστη με μορφή αναγνώσιμη από τον άνθρωπο δεν επιτρέπει την αυτόματη επεξεργασία.

Οι επίσημες γλώσσες πολιτικής (*formal policy languages*) έχουν σχεδιαστεί για να αντιπροσωπεύουν με σαφήνεια τις πολιτικές χρήσης, γεγονός που καθιστά δυνατή την αυτόματη επαλήθευση, εάν η επεξεργασία δεδομένων καλύπτεται από τη συγκατάθεση που δίνεται από τα άτομα στα οποία ανήκουν τα δεδομένα.

Εδώ, εξετάζουμε εν συντομία τις τρέχουσες εναλλακτικές λύσεις για τα εργαλεία συμμόρφωσης με τον ΓΚΠΔ.

Το [32] επεκτείνει το Prov-O και το P-Plan με έννοιες ειδικές για τον ΓΚΠΔ με στόχο την περιγραφή της προέλευσης των δεδομένων και τη χρήση του SPARQL για τη διαμόρφωση ερωτημάτων σχετικά με τη συμμόρφωση κατά τη διάρκεια του κύκλου ζωής των δεδομένων.

Το Γραφείο του Επιτρόπου Πληροφοριών (*Information Commissioner's Office-ICO*) στο Ηνωμένο Βασίλειο [33], η Microsoft [34] και το TrustArc [35] έχουν αναπτύξει εργαλεία ελέγχου συμμόρφωσης που επιτρέπουν στις εταιρείες να αξιολογούν τη συμμόρφωση των εφαρμογών και των διαδικασιών τους συμπληρώνοντας ένα προκαθορισμένο ερωτηματολόγιο. Αυτές οι προσεγγίσεις είναι χειροκίνητες όσον αφορά τον έλεγχο συμμόρφωσης..

Το GConsent [36] είναι μια οντολογία για το μοντέλο συναίνεσης που βασίζεται στο ΓΚΠΔ με δομημένο τρόπο, η οποία βρίσκεται υπό ανάπτυξη.

Ένα άλλο αξιοσημείωτο έργο είναι το Business Process Re-engineering and functional toolkit for GDPR compliance (BPR4GDPR) [37], ένα ερευνητικό project στα πλαίσια της Ε.Ε. Το project επικεντρώθηκε σε θέματα που βασίζονται σε διαδικασίες για τον έλεγχο των πολιτικών ασφάλειας και συμμόρφωσης, με αποτέλεσμα το Compliance Ontology [38].

Αν και δεν έχει σχεδιαστεί ρητά για τον ΓΚΠΔ, το PrivOnto [39] παρέχει ένα πλαίσιο που θα μπορούσε να διευκολύνει την ανάλυση των πολιτικών απορρήτου, συμπεριλαμβανομένης της συμμόρφωσης με την πολιτική χρήσης.

2.5 Επίλογος

Στο παρόν κεφάλαιο παρουσιάσαμε αναλυτικά αναφορές σε σχετική έρευνα που έχει πραγματοποιηθεί. Στο επόμενο κεφάλαιο αναλύουμε τις λειτουργικές και τις μη λειτουργικές απαιτήσεις του συστήματος μας, ενώ παραθέτουμε τα δομικά στοιχεία της αρχιτεκτονικής του με συνοδεία διαγραμμάτων.

Κεφάλαιο 3ο: Απαιτήσεις - Αρχιτεκτονική

3.1 Εισαγωγή

Σκοπός του παρόντος κεφαλαίου αποτελεί η περιγραφή των λειτουργικών και μη απαιτήσεων του συστήματος καθώς και της εννοιολογικής αρχιτεκτονικής του για ασφαλή ανάλυση δεδομένων με σεβασμό στην ιδιωτικότητα. Το κομμάτι της αρχιτεκτονικής θα ανταποκρίνεται στις απαιτήσεις που αποτυπώνονται στις επόμενες ενότητες. Τα κύρια θέματα της αρχιτεκτονικής θα είναι η παρουσίαση των τύπων χρηστών που αλληλεπιδρούν με το σύστημα και η ανάλυση του ρόλου τους. Έπειτα θα αναζητήσουμε και θα παραθέσουμε τα δομικά στοιχεία του συστήματος αρχικά σε γενικό επίπεδο και κατόπιν σε υποστοιχεία.

3.2 Απαιτήσεις

3.2.1 Λειτουργικές Απαιτήσεις

3.2.1.1 Απαιτήσεις για ασφαλή ανάλυση δεδομένων

3.2.1.1.1 Ανάλυση δεδομένων

Η πλατφόρμα πρέπει να επιτρέπει στους αναλυτές δεδομένων να πραγματοποιούν ανάλυση με σεβασμό στην ιδιωτικότητα χωρίς να αποκαλύπτουν μεμονωμένα δεδομένα του χρήστη. Για να μειωθεί η «επιφάνεια» επίθεσης, τα εισαχθέντα και τα ενδιάμεσα δεδομένα πέρα από την ανάλυση δεν πρέπει να διατηρούνται στην πλατφόρμα. Τα δεδομένα πρέπει να μεταφραστούν σε ένα προκαθορισμένο σχήμα δεδομένων (ουσιαστικά να μετασχηματιστούν), ώστε να μπορέσουν να χρησιμοποιηθούν στο *Ασφαλές Περιβάλλον Ανάλυσης Δεδομένων*. Ακολούθως θα πρέπει να είναι δυνατή η ανάλυση με δεδομένα: i) που προέρχονται από την ίδια εφαρμογή (την οποία παρέχει κάποιος οργανισμός) ii) που προέρχονται από διαφορετικές εφαρμογές (ο χρήστης χρησιμοποιεί πολλαπλές εφαρμογές από διαφορετικές εταιρείες) λ.χ. συσχέτιση του καρδιακού ρυθμού με το επίπεδο χοληστερόλης, παρόλο που το καθένα συλλέχθηκε από διαφορετική εφαρμογή. Ακόμη, όλοι οι υπολογισμοί ανάλυσης δεδομένων σε μεμονωμένα δεδομένα πρέπει να πραγματοποιούνται εντός της πλατφόρμας και τα περιβάλλοντα ανάλυσης μπορούν να αποθηκευτούν μόνο για περιορισμένο χρονικό διάστημα, όπως ορίζεται από την πολιτική διατήρησης δεδομένων της πλατφόρμας. Η πλατφόρμα πρέπει να διασφαλίζει ότι το περιβάλλον ανάλυσης που δημιουργείται για τον ερευνητή Ε είναι αποκλειστικά προσβάσιμο από αυτόν. Φυσικά η ανάλυση δεδομένων μπορεί να ξεκινήσει μόνο i) εάν ο αναλυτής συνδεθεί επιτυχώς στην πλατφόρμα και ii) τα δεδομένα που θα χρησιμοποιηθούν στην ανάλυση είναι επαρκή (μια μικρή ποσότητα δεδομένων μπορεί να οδηγήσει σε αποκάλυψη της ταυτότητας του χρήστη).

3.2.1.1.2 Διαχείριση πολιτικής χρήσης

Η ανάλυση δεδομένων απαιτεί ακριβή συγκατάθεση από τους εμπλεκόμενους χρήστες και ο έλεγχος πρόσβασης πρέπει να διασφαλίσει ότι μόνο εξουσιοδοτημένοι αναλυτές μπορούν να εργαστούν με τα δεδομένα. Για αυτό το λόγο, οι πολιτικές χρήσης, οι οποίες περιλαμβάνουν τόσο τη συγκατάθεση του χρήστη όσο και τη ρύθμιση του χειρισμού δεδομένων, πρέπει να εκπροσωπούνται σε μορφή αναγνώσιμη από μηχανή. Συν τοις άλλοις απαιτείται ένας μηχανισμός για τη διεξαγωγή αυτόματου ελέγχου πολιτικής χρήσης μεταξύ συγκατάθεσης και χειρισμού δεδομένων.

3.2.1.1.3 Δημοσίευση Δεδομένων

Οι αναλυτές δεν πρέπει να κατεβάζουν δεδομένα στην αρχική τους μορφή, ώστε να πληρούν τις απαιτήσεις και να ελαχιστοποιούν τις πιθανότητες για επιθέσεις απορρήτου, όπως η εκ νέου αναγνώριση του χρήστη. Αντί αυτού, για ανάλυση εκτός σύνδεσης, η πλατφόρμα θα πρέπει να προσφέρει λήψεις σε ανώνυμη μορφή και δημιουργία συνθετικών δεδομένων. Εκτός αυτού η πλατφόρμα πρέπει να παρέχει μέτρα ώστε να αποτρέπει την λήψη παρόμοιων δεδομένων πολλές φορές από έναν μόνο αναλυτή, καθώς αυτό μπορεί να αποκαλύψει ευαίσθητες πληροφορίες.

3.2.1.2 Απαιτήσεις για καταγραφή ελέγχου

Η προέλευση των δεδομένων πρέπει να εντοπίζεται σε ολόκληρο τον κύκλο ζωής τους εντός της πλατφόρμας. Η συλλογή που θα πραγματοποιείται αυτόματα, θα καταγράφει δεδομένα σε υψηλό επίπεδο προσδιορίζοντας ποιος, πότε και πώς έχουν προσπελαστεί τα δεδομένα καθώς και πότε υφίστανται επεξεργασία ή ανάλυση. Επιπλέον η καταγραφή προέλευσης πρέπει να περιλαμβάνει πληροφορίες σχετικά με το λογισμικό (όπως έκδοση) που χρησιμοποιείται για την επεξεργασία των δεδομένων. Φυσικά οι αναλυτές δεν πρέπει να είναι σε θέση να αποκτούν ή να συγκεντρώνουν προσωπικά δεδομένα μέσω αυτής της καταγραφής.

3.2.1.2.1 Διατήρηση μεταδεδομένων μετά τη διαγραφή

Οι χρήστες έχουν δικαίωμα να ανακαλέσουν την συγκατάθεση για οποιαδήποτε πληροφορία τους στην πλατφόρμα οποιαδήποτε στιγμή. Όταν η ανάκληση λάβει χώρα, η συγκατάθεση εξακολουθεί να ισχύει για προηγούμενες μελέτες. Προκειμένου να εκπληρωθεί η απαίτηση σωστής συλλογής και ανάλυσης δεδομένων, πρέπει να διατηρούνται κάποιες ελάχιστες απαραίτητες μετα-πληροφορίες για προηγούμενες μελέτες που διεξήχθησαν και βασίζονται σε διαγραμμένες πληροφορίες.

3.2.2 Μη λειτουργικές απαιτήσεις

Σε αυτήν την υποενότητα παραθέτουμε κάποιες μη λειτουργικές απαιτήσεις, οι οποίες είναι προδιαγραφές που κρίνουν τις δυνατότητες λειτουργίας και τους περιορισμούς του συστήματος, ώστε να ενισχύσουν τη λειτουργικότητά του.

3.2.2.1 Φορητότητα

Η φορητότητα ορίζει τον τρόπο με τον οποίο ένα σύστημα ή ένα στοιχείο του μπορεί να εκκινηθεί σε διαφορετικά περιβάλλοντα. Συνήθως περιλαμβάνει υλικό, λογισμικό ή άλλες προδιαγραφές πλατφόρμας χρήσης. Με απλά λόγια, καθορίζει πόσο καλά εκτελούνται οι ενέργειες που εκτελούνται μέσω μιας πλατφόρμας σε μια άλλη. Επίσης, ορίζει πόσο καλά είναι δυνατή η πρόσβαση και η αλληλεπίδραση των στοιχείων του συστήματος από δύο διαφορετικά περιβάλλοντα. Οι υπηρεσίες του συστήματος λοιπόν θα πρέπει να μπορούν να εκκινούν για παράδειγμα ανεξαρτήτως λειτουργικού συστήματος.

3.2.2.2 Επεκτασιμότητα

Η αρχιτεκτονική και η υλοποίηση θα πρέπει να παρουσιάζουν επεκτασιμότητα. Η ικανότητα αυτή θα επιτρέπει το χειρισμό αύξησης του φόρτου εργασίας (*workload*) χωρίς υποβάθμιση της απόδοσης ή να μεγεθύνει γρήγορα. Επίσης η μεγέθυνση της αρχιτεκτονικής θα μπορεί να φιλοξενήσει περισσότερους χρήστες, περισσότερες διαδικασίες, περισσότερες συναλλαγές και πρόσθετους κόμβους και υπηρεσίες καθώς αλλάζουν οι απαιτήσεις και καθώς το σύστημα εξελίσσεται για να καλύψει τις μελλοντικές

ανάγκες ενός οργανισμού. Τα υπάρχοντα συστήματα επεκτείνονται όσο το δυνατόν περισσότερο χωρίς να αντικατασταθούν. Η επεκτασιμότητα επηρεάζει άμεσα την αρχιτεκτονική καθώς και την επιλογή στοιχείων υλικού και λογισμικού συστήματος.

3.3 Αρχιτεκτονική

3.3.1 Σενάριο

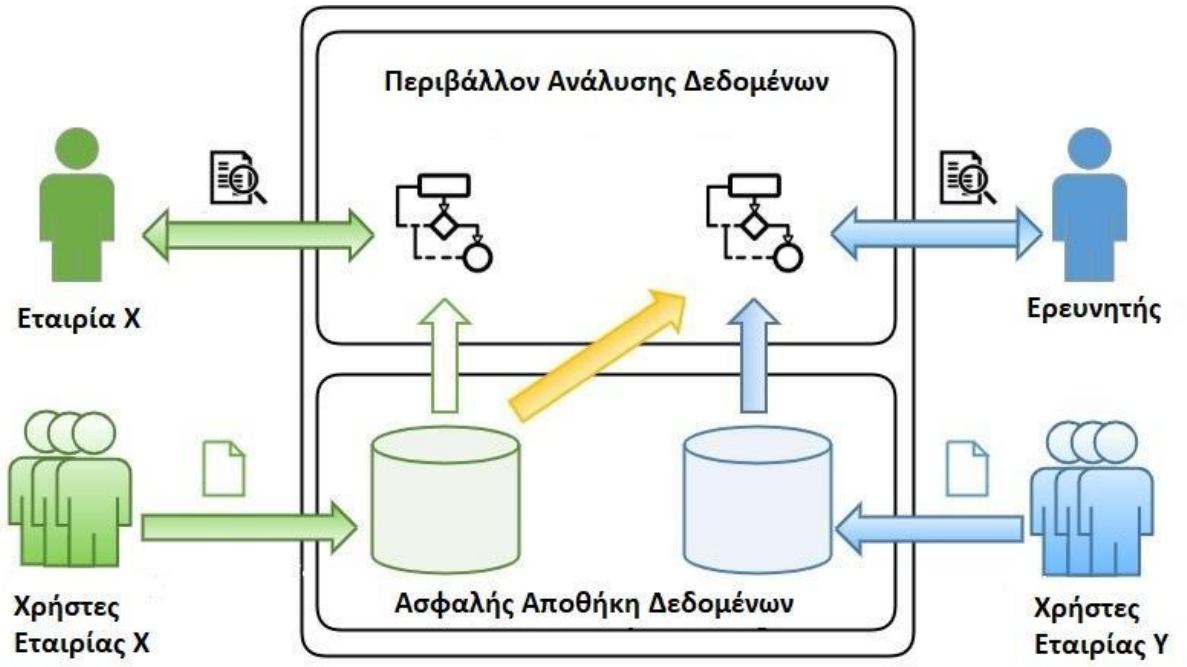
Στην παρούσα υποενότητα θα περιγράψουμε ένα σενάριο, σύμφωνα με το σχήμα 3.1, ώστε να διευκρινιστούν οι απαιτήσεις για την ελεγχόμενη ανάλυση δεδομένων, η οποία έχει πάντα ως γνώμονα τον σεβασμό της ιδιωτικότητας. Παρακάτω θα παρατεθούν 3 ΠΧ, βασισμένες σε αυτό το σενάριο για τη διεξαγωγή μελέτης σκοπιμότητας της προσέγγισής μας.

Έστω ότι ένα άτομο Α είναι χρήστης δύο εφαρμογών υγείας, μία από την εταιρία Χ, η οποία παρακολουθεί τον καρδιακό του ρυθμό και μία άλλη εταιρία Υ, η οποία καταγράφει το επίπεδο της χοληστερόλης του. Αφού ολοκληρωθεί η συλλογή δεδομένων και από τις δύο εφαρμογές, θέλει να συνδέσει και να επιτρέψει τον διαμοιρασμό των δεδομένων, αλλά μόνο για ερευνητικούς σκοπούς. Επιπρόσθετα λαμβάνει υπόψιν να κοινοποιήσει τα δεδομένα του για συγκεκριμένους εμπορικούς σκοπούς, όπως το να λάβει πρόταση προϊόντος, αλλά θέλει να το αποφασίσει αργότερα.

Ωστόσο, δεδομένου ότι και οι δύο εταιρείες είναι μικρές σε μέγεθος, δεν επιθυμούν να διαχειρίζονται τα δεδομένα από μόνες τους. Σκοπεύουν όμως να αποθηκεύσουν τα δεδομένα των χρηστών τους σε μια πλατφόρμα νέφους χωρίς να χρειάζεται να την αναπτύξουν.

Ένας ερευνητής Ε σχεδιάζει να αναπτύξει ένα μοντέλο μηχανικής μάθησης, το οποίο θα αναλύει τη συσχέτιση του καρδιακού ρυθμού με το επίπεδο χοληστερόλης για την έρευνά του. Ψάχνει τα κατάλληλα δεδομένα για το μοντέλο του, παράλληλα όμως θέλει να βεβαιωθεί ότι συμμορφώνεται με τους κανονισμούς. Αργότερα το άτομο Α ρωτάει το άτομο Δ, το οποίο ανήκει στους διαχειριστές της πλατφόρμας νέφους, σχετικά με τη χρήση των δεδομένων του. Καθώς όμως τα δεδομένα του χρήστη Α χρησιμοποιούνται ως κομμάτι της έρευνας του Ε, ο διαχειριστής αυτός παρέχει στον Α τις πληροφορίες σχετικά με το πείραμα που πρόκειται ή έχει ήδη διεξαχθεί.

Σε αυτό το σενάριο τα συλλεχθέντα δεδομένα (από τις προαναφερθείσες εταιρίες Χ και Υ) μπορούν να κατηγοριοποιηθούν ως ειδικά προσωπικά δεδομένα, σύμφωνα με το αρθ. 9 του ΓΚΠΔ. Η πλατφόρμα λοιπόν θα πρέπει να διευκολύνει χρήστες όπως ο Α, να καθορίζουν ρητή συγκατάθεση για τα δεδομένα τους. Επιπλέον για λόγους απορρήτου, οι πάροχοι εφαρμογών και οι ερευνητές δεν θα πρέπει να έχουν άμεση πρόσβαση στα δεδομένα του χρήστη, αλλά θα πρέπει να μπορούν να διεξάγουν ανάλυση με σεβασμό στην ιδιωτικότητα με τέτοιο τρόπο, ούτως ώστε να μην διακυβεύεται η συγκατάθεση του χρήστη. Το άτομο Δ από την άλλη πλευρά θα πρέπει να παρέχει πληροφορίες ελέγχου (όπως πλήθος πειραμάτων στα οποία έχουν χρησιμοποιηθεί τα δεδομένα) χωρίς να έχει πρόσβαση στο περιεχόμενο των προσωπικών δεδομένων του ατόμου Α.



Σχήμα 3.1: Απεικόνιση προσέγγισης

3.3.2 Χρήστες

Το πρώτο βήμα της σχεδίασης της αρχιτεκτονικής αποτελεί η εύρεση των χρηστών (*actors*) που θα αλληλεπιδρούν με το σύστημα. Σύμφωνα με τις προαναφερθείσες απαιτήσεις, μπορούμε να αναγνωρίσουμε τρεις διαφορετικούς τύπους χρηστών στην πλατφόρμα.

- *Τυπικοί χρήστες / Ιδιοκτήτες Δεδομένων*

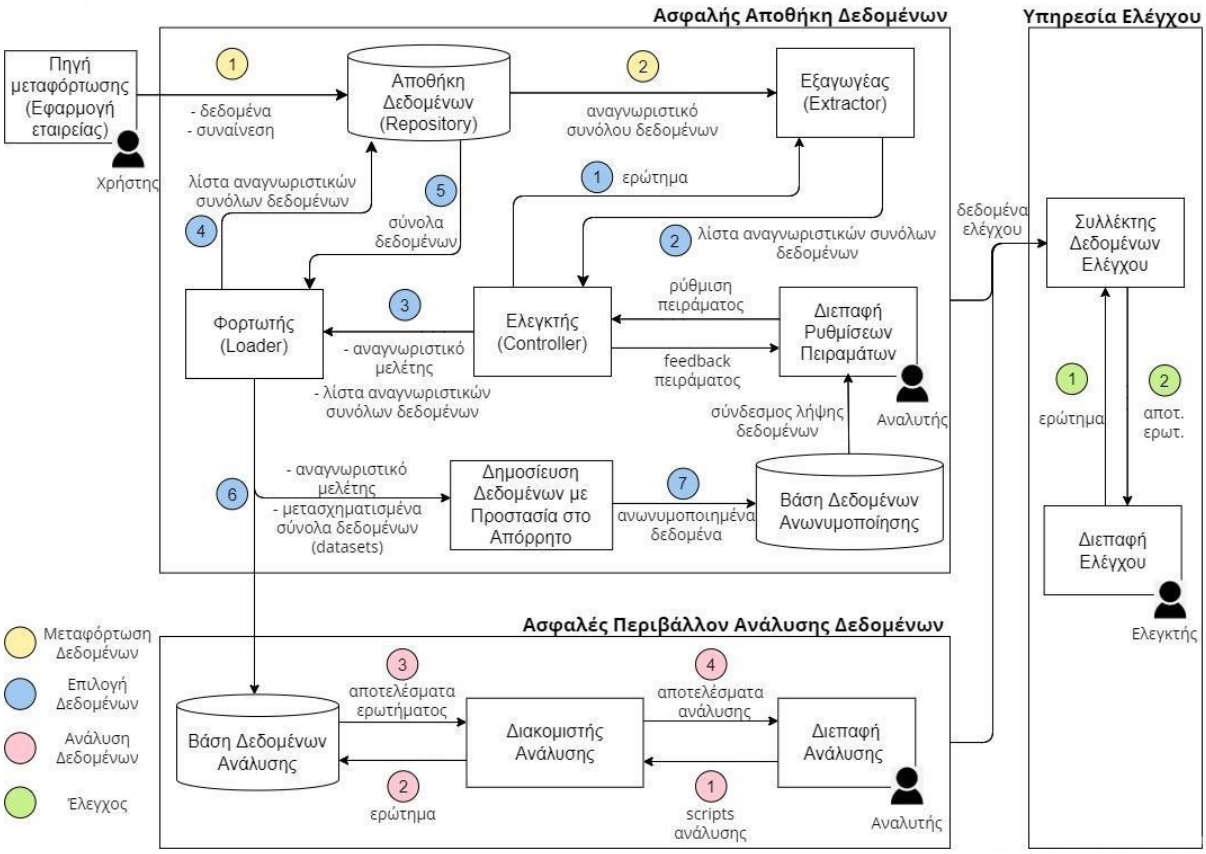
  Οι χρήστες στους οποίους ανήκουν τα δεδομένα που αποθηκεύονται στην πλατφόρμα. Μπορούν να δώσουν συγκατάθεση για την ανάλυση των δεδομένων τους. Η συγκατάθεσή τους μπορεί να εφαρμοστεί είτε σε ομάδες δεδομένων που τους ανήκουν είτε μεμονωμένα. Χρησιμοποιούν μια εφαρμογή που παρέχεται από τον αντίστοιχο οργανισμό ενώ δεν αλληλεπιδρούν άμεσα με την πλατφόρμα, όπως μέσω μιας ειδικής διεπαφής χρήστη. Οι εφαρμογές στις οποίες υπάρχει άμεση αλληλεπίδραση με αυτόν τον τύπο χρηστών είναι εκτός βεληνεκούς της παρούσας και δεν θα αναλυθούν περαιτέρω.

- *Αναλυτές*

  Οι χρήστες οι οποίοι μπορούν να διεξάγουν πειράματα στην πλατφόρμα. Καθορίζουν τους τύπους δεδομένων που θα χρησιμοποιηθούν στην ανάλυση και πραγματοποιούν την ανάλυση πάνω στα επιλεχθέντα δεδομένα.

- *Διαχειριστές / Ελεγκτές*

  Οι χρήστες οι οποίοι μπορούν να αναλύσουν αποδεικτικά στοιχεία που συλλέγονται στην πλατφόρμα. Τα δεδομένα που συλλέγουν μπορούν να χρησιμοποιηθούν για να απαντήσουν σε συγκεκριμένες ερωτήσεις ελέγχου, π.χ. μια δικαστική υπόθεση ή απλές στατιστικές χρήσεις για χρήστες που θέλουν να γνωρίζουν πότε και από ποιον χρησιμοποιήθηκαν τα δεδομένα τους.



Σχήμα 3.2: Απεικόνιση εννοιολογικής αρχιτεκτονικής

3.3.3 Συστατικά στοιχεία

Για την αναζήτηση των δομικών στοιχείων της αρχιτεκτονικής, χρειαζόμαστε ένα συστατικό για κάθε βασική κατηγορία λειτουργικής απαίτησης, καθώς και ένα συστατικό για την αποθήκευση των δεδομένων. Συνεπώς η αρχιτεκτονική θα αποτελείται από τρεις ομάδες συστατικών, η καθεμία από τις οποίες εξυπηρετεί διαφορετικό σκοπό (βλ. σχήμα 3.2):

- *Ασφαλής Αποθήκη Δεδομένων (Repository)*

  Αποθηκεύει δεδομένα που έχουν μεταφορτωθεί από έναν χρήστη και επιτρέπει την επιλογή των δεδομένων που θα χρησιμοποιηθούν σε πειράματα από τον αναλυτή. Δεν υπάρχουν στοιχεία σε αυτήν την ομάδα που να επιτρέπουν στον αναλυτή να προβάλλει ή να αναλύει απευθείας ανεπεξέργαστα δεδομένα.

- *Ασφαλές Περιβάλλον Ανάλυσης Δεδομένων*

  Λαμβάνει επιλεγμένα μετασχηματισμένα δεδομένα που πληρούν κριτήρια πειράματος, π.χ. συγκατάθεση, όπου αντιγράφονται σε αυτό το στοιχείο για περαιτέρω ανάλυση. Αυτό το τμήμα της πλατφόρμας παρέχει μηχανισμούς που επιτρέπουν στους αναλυτές να διεξάγουν τα πειράματά τους με τρόπο διατήρησης του απορρήτου. Η επιλογή δεδομένων εκφράζεται συνήθως μέσω ερωτημάτων.

- *Υπηρεσία Ελέγχου*

  Συλλέγει και διαχειρίζεται δεδομένα προέλευσης για να υποστηρίξει τη δυνατότητα ελέγχου των διεργασιών εντός της *Ασφαλούς Αποθήκης Δεδομένων* και του *Ασφαλούς Περιβάλλοντος*

Ανάλυσης Δεδομένων. Παρέχει διεπαφή προς χρήση από τους ελεγκτές για τη δυνατότητα απάντησης ερωτήσεων που σχετίζονται με τον έλεγχο σχετικά με την πρόσβαση και τη χρήση προσωπικών δεδομένων.

3.3.4 Διεργασίες

Το σχήμα 3.2 απεικονίζει την εννοιολογική αρχιτεκτονική του συστήματος, στην οποία χρησιμοποιούμε τα εξωτερικά ορθογώνια για να υποδείξουμε τη διαίρεση των τριών αυτών δομικών στοιχείων. Φυσικά κάθε ομάδα απαρτίζεται από ορισμένα υποστοιχεία, τα οποία επικοινωνούν μεταξύ τους, ενώ κάποια μπορούν να επικοινωνούν και με υποστοιχεία άλλης ομάδας. Τα εικονίδια χρηστών αντιπροσωπεύουν την δυνατότητα αλληλεπίδρασης ενός τύπου χρήστη με το εκάστοτε υποστοιχείο. Επιπρόσθετα, στο σχήμα απεικονίζονται τέσσερις τυπικές διεργασίες που εκτελούνται στην πλατφόρμα:

- *Μεταφόρτωση δεδομένων χρήστη*

  Η διεργασία ξεκινάει όταν η εφαρμογή ενός χρήστη στέλνει δεδομένα στην πλατφόρμα. Κατά τη διάρκεια της διεργασίας αυτής, εξάγονται μεταδεδομένα από τα δεδομένα. Καθώς τα δεδομένα θα είναι σε μορφή αρχείων, θα πρέπει να υπάρχει μηχανισμός ανάγνωσης των περιεχομένων τους, ώστε να καθίσταται δυνατή η εξαγωγή. Κατόπιν τα μεταδεδομένα αποθηκεύονται μαζί με τα δεδομένα και τη συγκατάθεση που υποδεικνύεται κατά τη μεταφόρτωση στην πλατφόρμα. Έτσι, κάθε σύνολο δεδομένων που αποστέλλεται στην πλατφόρμα συνδέεται με ένα ελάχιστο σύνολο πληροφοριών που επιτρέπει την ανάκτησή του.

- *Επιλογή δεδομένων*

  Κατά τη διεργασία αυτή, οι αναλυτές ορίζουν τα κριτήρια αναζήτησης για τα δεδομένα που θέλουν να χρησιμοποιήσουν στα πειράματά τους. Εάν η πλατφόρμα έχει αρκετά δεδομένα που πληρούν τα κριτήρια αυτής της αναζήτησης (επίσης και για τη συγκατάθεση χρήστη), τότε η διεργασία φορτώνει τα πραγματικά δεδομένα (όπως αυτά έχουν μεταφορτωθεί) στο ΑΠΑΔ. Σε αυτό ο αναλυτής λαμβάνει ακριβή αποτελέσματα σε συγκεντρωτική μορφή, χωρίς να έχει πρόσβαση σε μεμονωμένα σύνολα δεδομένων. Η αναζήτηση δεδομένων βασίζεται αποκλειστικά σε πληροφορίες υψηλού επιπέδου που παρέχονται στα μεταδεδομένα. Για παράδειγμα, η *Γλυκόζη Αίματος*, η οποία είναι ένα από τα χαρακτηριστικά των πραγματικών δεδομένων χρήστη, αποθηκεύεται ως αλφαριθμητική τιμή σε μια λίστα στα αντίστοιχα μεταδεδομένα, η οποία υποδεικνύει ποια χαρακτηριστικά είναι παρόντα στο συνδεδεμένο αρχείο δεδομένων. Επιπλέον, οι αναλυτές μπορούν να κατεβάσουν ανωνυμοποιημένα δεδομένα και να τα αναλύσουν εκτός της πλατφόρμας (offline). Ωστόσο, η ανωνυμοποίηση μπορεί να μειώσει σημαντικά την ποιότητα των δεδομένων και τα αποτελέσματα της ανάλυσης των δεδομένων εκτός σύνδεσης.

- *Ανάλυση δεδομένων*

  Στην διεργασία αυτή, οι αναλυτές επεξεργάζονται τα δεδομένα και παράγουν αποτελέσματα υποβάλλοντας κώδικα στην πλατφόρμα. Η πλατφόρμα πρέπει να διασφαλίζει ότι οι αναλυτές δεν θα είναι σε θέση να εντοπίσουν ή να συμπεράνουν τα υποκείμενα των δεδομένων από την ανάλυση.

- *Έλεγχος*

Κατά τη διεργασία αυτή, οι πληροφορίες προέλευσης λαμβάνονται από τα διάφορα υποστοιχεία της πλατφόρμας και συγκεντρώνονται σε έναν κεντρικό συλλέκτη, όπου οι ελεγκτές μπορούν να προβάλλουν τις λεπτομέρειες πειραμάτων.

Τα χρώματα των κύκλων υποδεικνύουν την διεργασία στην οποία ανήκει η δράση (*action*) που εκτελείται: κίτρινο (Μεταφόρτωση Δεδομένων Χρήστη), μπλε (Επιλογή Δεδομένων), ροζ (Ανάλυση Δεδομένων) και πράσινο (Έλεγχος). Οι αριθμοί υποδηλώνουν την σειρά με την οποία λαμβάνουν χώρα οι δράσεις της αντίστοιχης διεργασίας. Σε αυτό το σημείο περιγράφουμε κάθε μία από τις δράσεις λεπτομερώς για να εξηγήσουμε τον ρόλο καθενός από τα συστατικά.

3.3.4.1 Μεταφόρτωση Δεδομένων Χρήστη

Στην διεργασία αυτή, τα δεδομένα που ανεβάζουν οι χρήστες μετασχηματίζονται και εμπλουτίζονται, ώστε να μπορούν αργότερα να εντοπιστούν και να προσπελαστούν σύμφωνα με την αντίστοιχη συναίνεση χρήστη. Η διεργασία ξεκινά όταν η εφαρμογή ενός χρήστη στέλνει δεδομένα μαζί με τη συγκατάθεση στο *Ασφαλές Αποθετήριο Δεδομένων*. Ο κύριος τρόπος αλληλεπίδρασης είναι μέσω μιας Διεπαφής Προγραμματισμού Εφαρμογών *(API)*, όπου μια κλήση σε μια ορισμένη διαδρομή θα πυροδοτήσει τη μεταφόρτωση (βήμα 1). Το *Αποθετήριο Δεδομένων* δημιουργεί μια εγγραφή στην οποία αποθηκεύει το μόνιμο αναγνωριστικό (*Persistent Identifier/PID*), τη συγκατάθεση και τη θέση των δεδομένων που λαμβάνονται. Ένα PID είναι μια "μακροχρόνια" αναφορά σε έναν ψηφιακό πόρο. Ουσιαστικά είναι μια ετικέτα που δίνει ένα μοναδικό όνομα σε μια οντότητα: ένα άτομο, ένα μέρος ή ένα πράγμα (στην περίπτωσή μας στις εγγραφές του αποθετηρίου). Σε αντίθεση με τις διευθύνσεις ΕΕΠ (*URI*), οι οποίες ενδέχεται να "σπάσουν", ένα PID οδηγεί αξιόπιστα σε μια ψηφιακή οντότητα. Το PID είναι μοναδικό και δεν αλλάζει ποτέ. Ακολουθώντας τις αρχές FAIR (*FAIR Principles*) [40],[41], οι οποίες είναι η *Εύρεση*, η *Προσβασιμότητα*, η *Διαλειτουργικότητα* και η *Επαναχρησιμοποίηση (Findability, Accessibility, Interoperability, Reusability)*, ακόμα κι αν τα δεδομένα διαγραφούν, το PID διατηρείται και το αρχείο ενημερώνεται με πληροφορίες σχετικά με το γιατί και πότε αφαιρέθηκαν τα δεδομένα.

Η πλατφόρμα δημιουργεί αυτόματα μεταδεδομένα και τα αποθηκεύει μαζί με τη συγκατάθεση στον Εξαγωγέα (Extractor) (βήμα 2). Το στοιχείο αυτό λαμβάνει το PID του συνόλου δεδομένων και έπειτα λαμβάνει πρόσβαση στα δεδομένα. Φυσικά κάποιες πληροφορίες για την εγγραφή αποθηκεύονται στην βάση δεδομένων (βήμα 3), όπως η συναίνεση και ορισμένα μεταδεδομένα που υπάρχουν στο αποθετήριο, συμπεριλαμβανομένου του PID (όπου χρησιμοποιείται αργότερα για ανάκτηση των εγγραφών και κατά συνέπεια των αντίστοιχων δεδομένων).

3.3.4.2 Επιλογή Δεδομένων

Οι αναλυτές χρησιμοποιούν την *Διεπαφή Ρυθμίσεων Πειραμάτων* για να ορίσουν τα πειράματά τους. Καθορίζουν τον σκοπό του πειράματος και ποια χαρακτηριστικά δεδομένων θέλουν να χρησιμοποιήσουν στην ανάλυσή τους. Με βάση αυτήν την είσοδο, ο *Ελεγκτής (Controller)* δημιουργεί αυτόματα ένα ερώτημα SQL και το στέλνει στη *Βάση Δεδομένων* (βήμα 1). Η βάση επιστρέφει μια λίστα με μοναδικά αναγνωριστικά (PID) δεδομένων που πληρούν τα κριτήρια (βήμα 2). Εάν ο αριθμός των διαθέσιμων συνόλων δεδομένων πληροί τις απαιτήσεις του αναλυτή (έτσι ώστε η μελέτη να έχει νόημα από στατιστική άποψη), μπορεί να αποφασίσει να ξεκινήσει μια νέα μελέτη και να φορτώσει τα δεδομένα στο ΑΠΑΔ.

Κάθε μελέτη έχει το δικό της PID και συνδέσμους με τη λίστα των PID δεδομένων, καθώς και μεταδεδομένα σχετικά με το πείραμα. Παρόμοια με τα PID που χρησιμοποιούνται για σύνολα δεδομένων, το PID της μελέτης μας επιτρέπει να αναφερόμαστε σε πειράματα με ξεκάθαρο τρόπο.

Ο *Φορτωτής* (*Loader*) λαμβάνει το PID της μελέτης (βήμα 3) και ανακτά τα δεδομένα από το *Αποθετήριο Δεδομένων* (βήμα 5), αφού παράσχει τη λίστα με τα PID σε αυτό (βήμα 4). Ανάλογα με την υλοποίηση του ΑΠΑΔ, ο *Φορτωτής* μετατρέπει τα δεδομένα σε κατάλληλη μορφή, π.χ. βάση δεδομένων, συγκεντρωτικούς πίνακες κ.λπ (βήμα 6). Όταν φορτωθούν τα δεδομένα, οι αναλυτές λαμβάνουν πληροφορίες μέσω της *Διεπαφής Ρυθμίσεων Πειραμάτων* για τον τρόπο πρόσβασης στο ΑΠΑΔ, π.χ. μια διεύθυνση URL για πρόσβαση, όνομα χρήστη και κωδικό πρόσβασης. Στο σχήμα 3.2, τα βέλη αυτού του βήματος σχηματίζουν διχάλα, καθώς οι αναλυτές μπορούν να αποφασίσουν να πραγματοποιήσουν λήψη είτε ανωνυμοποιημένων δεδομένων είτε συνθετικών δεδομένων αντί ανάλυσης στο χώρο της πλατφόρμας. Για να το κάνουν αυτό, χρησιμοποιούν τη *Διεπαφή Ρυθμίσεων Πειραμάτων* για να ορίσουν το πείραμά τους. Όλα τα βήματα είναι τα ίδια με πριν, ωστόσο τα δεδομένα δεν φορτώνονται στη *Βάση Δεδομένων Ανάλυσης Δεδομένων* αλλά μετασχηματίζονται από το στοιχείο *Δημοσίευσης Δεδομένων με Προστασία στο Απόρρητο* και διατίθενται για λήψη.

3.3.4.3 Ανάλυση Δεδομένων

Οι αναλυτές χρησιμοποιούν τη *Διεπαφή Ανάλυσης* για να επικοινωνήσουν με το *Διακομιστή Ανάλυσης* (βήμα 1), ο οποίος με τη σειρά του έχει πρόσβαση στα δεδομένα που είναι αποθηκευμένα στη *Βάση Δεδομένων Ανάλυσης* (βήμα 2). Η *Διεπαφή Ανάλυσης* επιτρέπει την υποβολή κώδικα για ανάλυση δεδομένων και την προβολή των αποτελεσμάτων επεξεργασίας. Καταγράφει επίσης κάθε δραστηριότητα του αναλυτή. Η διεπαφή μπορεί να υλοποιηθεί ως εφαρμογή Ιστού που εκτελείται από την πλευρά του διακομιστή.

Ο *Διακομιστής Ανάλυσης* ελέγχει τον κώδικα που υποβάλλουν οι αναλυτές και προστατεύει τα δεδομένα από άμεση πρόσβαση. Για παράδειγμα, ο διακομιστής επιτρέπει την εκτέλεση μόνο συναρτήσεων για συγκέντρωση *(aggregation)* και αποκλείει αιτήματα για την εμφάνιση ιδιοτήτων συγκεκριμένων συνόλων δεδομένων (όπως χρήστες). Εάν τα υποβληθέντα scripts πληρούν αυτές τις απαιτήσεις για τη διατήρηση του απορρήτου, τότε τα αποτελέσματα υπολογίζονται στα αρχικά δεδομένα και επιστρέφονται στη *Διεπαφή Ανάλυσης*.

Σημειώνουμε ότι η *Διεπαφή Ανάλυσης* είναι μέρος της πλατφόρμας και δεν είναι δυνατή η άμεση πρόσβαση στο *Διακομιστή Ανάλυσης*. Ένας τέτοιος σχεδιασμός είναι πιο ασφαλής και επιτρέπει την καλύτερη παρακολούθηση της διαδικασίας ανάλυσης.

Μια διεργασία ανάλυσης δεδομένων μπορεί να εκτελεστεί μόνο όταν τα δεδομένα έχουν φορτωθεί στο ΑΠΑΔ και όταν οι αναλυτές λάβουν τα στοιχεία σύνδεσής τους. Η *Βάση Δεδομένων Ανάλυσης Δεδομένων* λαμβάνει μόνο τα δεδομένα που απαιτούνται για τη συγκεκριμένη μελέτη, τα διατηρεί για περιορισμένο χρονικό διάστημα και τα διαγράφει μετά την ολοκλήρωση της μελέτης. Κατά τη διάρκεια της ανάλυσης, δεν υπάρχει ανάγκη πρόσβασης στο *Αποθετήριο Δεδομένων*. Ως εκ τούτου, χάρη στη χαλαρή σύζευξη μεταξύ των ομάδων στοιχείων (δηλ. τα ορθογώνια, βλ. σχήμα 3.2), κάθε τέτοια ομάδα μπορεί να δημιουργηθεί με την ανάπτυξη *(deployment)* της σε διαφορετικούς κόμβους. Αυτό συνεπάγεται σε διαφορετικά/καλύτερα επίπεδα ασφάλειας, πρόσβασης, καθώς και υπολογιστικής ισχύος.

3.3.4.4 Έλεγχος

Η *Ασφαλής Αποθήκη Δεδομένων* και το *Ασφαλές Περιβάλλον Ανάλυσης Δεδομένων* καταγράφουν πληροφορίες προέλευσης και τις προωθούν στο *Συλλέκτη Δεδομένων Ελέγχου*. Ο σκοπός εδώ είναι ο διαχωρισμός αυτών των πληροφοριών από τα δεδομένα επεξεργασίας στοιχείων, στα οποία καταγράφηκε η προέλευση. Ο σχεδιασμός αυτός ενισχύει την ασφάλεια των πληροφοριών προέλευσης, επειδή αποτρέπει την τροποποίησή τους χωρίς εξουσιοδότηση όταν ένα από τα στοιχεία, που επιτρέπουν άμεση αλληλεπίδραση με το χρήστη, εκτεθεί σε κάποιο κίνδυνο. Η καταγραφή προέλευσης μπορεί να υλοποιηθεί με διάφορους τρόπους.

Ο *Συλλέκτης Δεδομένων Ελέγχου* αποθηκεύει τις πληροφορίες προέλευσης για καθεμία από τις διεργασίες που υποστηρίζονται από την πλατφόρμα. Πιο συγκεκριμένα:

- Μεταφόρτωση Δεδομένων Χρήστη

  Αποθηκεύεται το PID ενός συνόλου δεδομένων, η συναίνεση, τα μεταδεδομένα που δημιουργούνται αυτόματα και συνοψίζουν τα περιεχόμενα του συνόλου δεδομένων, μια χρονική σήμανση.

- Επιλογή Δεδομένων

  Αποθηκεύεται το αναγνωριστικό του αναλυτή, ένα αναγνωριστικό του υποβληθέντος ερωτήματος, χαρακτηριστικά ερωτήματος (π.χ. χρονική σήμανση, σκοπός μελέτης, περιγραφή μελέτης), αποτελέσματα ερωτήματος, αποτελέσματα ελέγχου συναίνεσης και ένα αναγνωριστικό της ανάλυσης (μόνο εάν ο έλεγχος συγκατάθεσης ήταν επιτυχής και ο αναλυτής αποφασίσει να αναλύσει τα δεδομένα).

- Ανάλυση Δεδομένων

  Αποθηκεύεται ένα αναγνωριστικό της ανάλυσης (που δημιουργήθηκε στο προηγούμενο βήμα για τη σύνδεση του ερωτήματος με την ανάλυση), πληροφορίες για το περιβάλλον λογισμικού και τις εξαρτήσεις (π.χ. λειτουργικό σύστημα, βιβλιοθήκες λογισμικού, μεταβλητές περιβάλλοντος), παραμέτρους σεναρίου , αποτελέσματα της ανάλυσης.

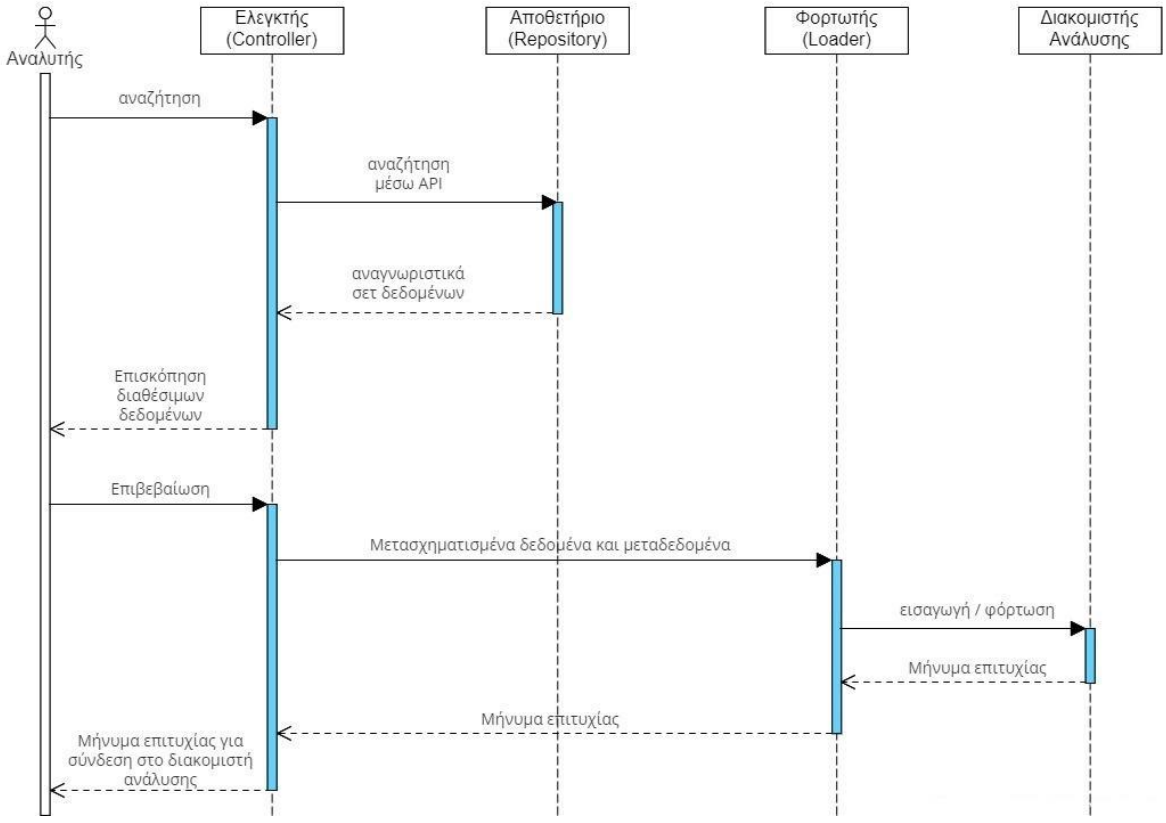
  Οι ελεγκτές χρησιμοποιούν τη *Διεπαφή Ελέγχου* για να αποκτήσουν πρόσβαση σε πληροφορίες προέλευσης που οργανώνονται στο *Συλλέκτη Δεδομένων*. Μπορούν να προβάλλουν μια επισκόπηση πρόσφατων ερευνών ή να ορίσουν πιο εξειδικευμένα ερωτήματα για την ανάκτηση συγκεκριμένων πληροφοριών προέλευσης κατά την εκτέλεση ενός ελέγχου (*audit*).

3.3.5 Διαγράμματα

Στην παρούσα υποενότητα παραθέτουμε διαγράμματα ακολουθίας καθώς και ένα απλό σχήμα της βάσης δεδομένων, τα οποία θα αποτελέσουν οδηγό για το κεφάλαιο υλοποίησης. Τονίζουμε πως οι απεικονιζόμενες διεργασίες στα διαγράμματα αυτά, έχουν ως περιγραφή περιεκτικές φράσεις, παρά ονόματα μεθόδων ή συναρτήσεων, όπως είναι το σύνηθες πρότυπο. Αυτό γίνεται για καλύτερη κατανόηση από τον αναγνώστη. Το σχήμα 3.3 απεικονίζει το διάγραμμα ροής για την επιλογή δεδομένων από τον αναλυτή. Φυσικά το διάγραμμα αυτό συνάδει με μέρος των διεργασιών που απεικονίζονται στο σχήμα 3.2. Τα στοιχεία στα οποία λαμβάνουν χώρα οι διαδικασίες για την επιλογή δεδομένων προς ανάλυση είναι ο Ελεγκτής, το Αποθετήριο, ο Φορτωτής και ο Διακομιστής Ανάλυσης. Η επιλογή δεδομένων εκκινεί με την αναζήτηση του αναλυτή για δεδομένα επιλέγοντας φίλτρα συναίνεσης, λεπτομέρειες πειράματος και χαρακτηριστικά προς ανάλυση (π.χ. Πίεση Αίματος).

Σημειώνουμε ότι στο σχήμα δεν περιλαμβάνεται η αυθεντικοποίηση του χρήστη με την πλατφόρμα, η οποία είναι προϋπόθεση για όλες τις διαδικασίες. Ο *Ελεγκτής* επικυρώνει τα εισερχόμενα δεδομένα που έχουν υποβληθεί από τον αναλυτή. Κατόπιν πραγματοποιεί μια κλήση μέσω ΔΠΕ στο *Αποθετήριο Δεδομένων*, όπου τα φίλτρα μπορούν να ταιριάξουν κανένα ή περισσότερα δεδομένα. Κατόπιν, το *Αποθετήριο Δεδομένων* επιστρέφει ως απάντηση μια λίστα με τα PID στον *Ελεγκτή*. Αφού πραγματοποιηθεί έλεγχος για επαρκή δεδομένα (πρέπει να τεθεί ένα όριο για ένα ελάχιστο πλήθος των δεδομένων που πληροί τα κριτήρια αναζήτησης), τότε ο ελεγκτής εμφανίζει την επισκόπηση (στην ουσία τον αριθμό των διαθέσιμων συνόλων δεδομένων) στον αναλυτή. Ο αναλυτής μπορεί να αποφασίσει αν ο αριθμός αυτός είναι ικανοποιητικός για την επικείμενη έρευνα. Στην περίπτωση επιβεβαίωσης, τα μετασχηματισμένα δεδομένα και ορισμένα μεταδεδομένα από το *Αποθετήριο Δεδομένων*, μεταφέρονται στο *Φορτωτή*. Ο *Φορτωτής* προετοιμάζει μεταδεδομένα που προορίζονται για το *Διακομιστή Ανάλυσης* και τα στέλνει μαζί με τα μετασχηματισμένα δεδομένα και το αναγνωριστικό μελέτης σε αυτόν. Εάν η φόρτωση είναι επιτυχής, υπάρχει η αλυσίδα από μηνύματα επιτυχίας κατά σειρά: *Διακομιστής Ανάλυσης* - *Φορτωτής* - *Ελεγκτής* - *Αναλυτής*.

Διάγραμμα Ροής #1: Ο αναλυτής δεδομένων ψάχνει για δεδομένα. Τα επιλεγμένα δεδομένα φορτώνονται στον διακομιστή ανάλυσης. Απεικόνιση χωρίς αυθεντικοποίηση.

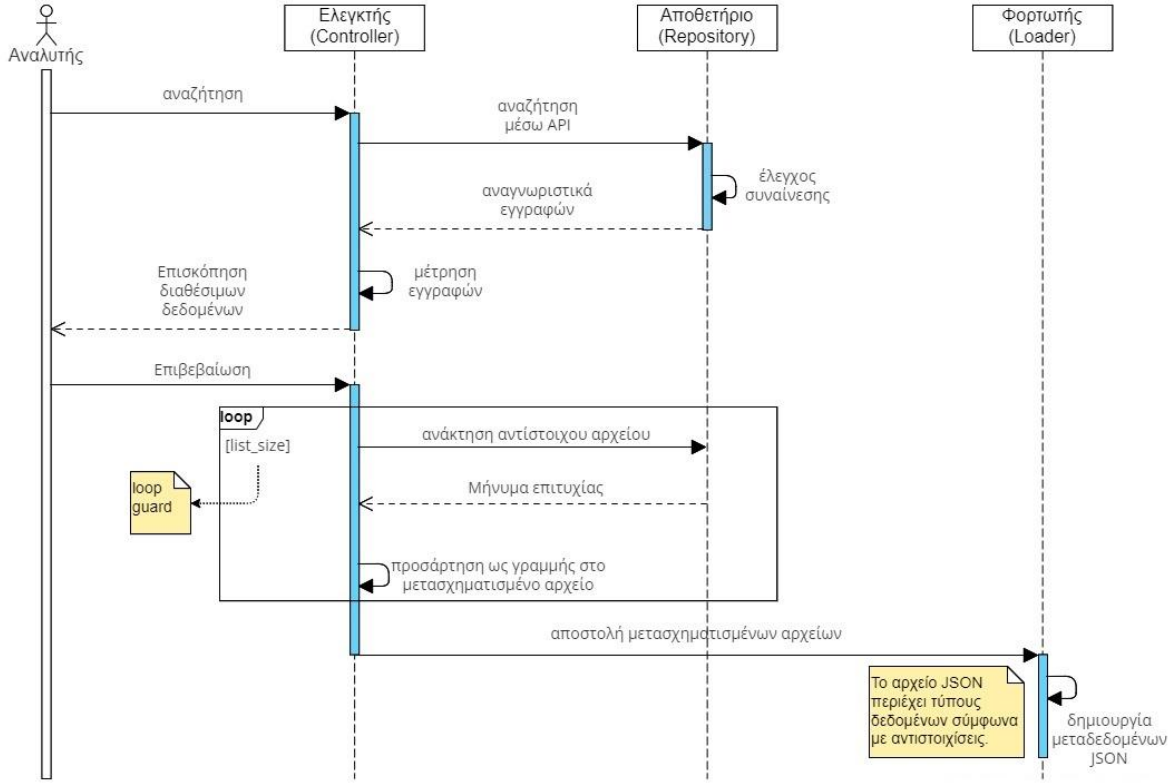


Σχήμα 3.3: Απεικόνιση ακολουθίας αναζήτησης δεδομένων

Το σχήμα 3.4 αποτελεί μια πιο λεπτομερή έκδοση του προηγούμενου διαγράμματος ακολουθίας (σχήμα 3.3). Σε αυτό περιγράφουμε την επιλογή δεδομένων από τον αναλυτή, παραθέτοντας περισσότερα γεγονότα που λαμβάνουν χώρα κατά τη διεργασία. Η ακολουθία έχει την ίδια αφετηρία, αλλά σταματάει πριν την φόρτωση των δεδομένων από τον *Φορτωτή* στον *Διακομιστή Ανάλυσης*. Το *Αποθετήριο*

Δεδομένων πραγματοποιεί έλεγχο για την συναίνεση που έχει δοθεί από τον Αναλυτή κοιτώντας τα μεταδεδομένα από τα αρχεία δεδομένων. Για κάθε αρχείο του οποίου τα μεταδεδομένα ταιριάζουν στα φίλτρα συναίνεσης, το *Αποθετήριο* προσαρτεί το αντίστοιχο PID σε μία ενιαία λίστα, η οποία θα επιστραφεί στον *Ελεγκτή* στο τέλος του ελέγχου συναίνεσης. Ο *Ελεγκτής* καταμετράει το μέγεθος της επιστρεφόμενης λίστας και επιστρέφει τον αριθμό στην επισκόπηση διαθέσιμων δεδομένων. Σημειώνουμε ότι η μέτρηση γίνεται στον *Ελεγκτή* και όχι στο *Αποθετήριο*, καθώς ο πρώτος χρειάζεται τα PID για να ανακτήσει τα περιεχόμενα κάθε αρχείου στην περίπτωση επιβεβαίωσης. Κατόπιν της επιβεβαίωσης λοιπόν από τον αναλυτή, απεικονίζεται μια επανάληψη η οποία εξυπηρετεί το σκοπό του μετασχηματισμού των αρχείων. Η λογική είναι απλή: Για κάθε PID στη λίστα, ανακτάται το αντίστοιχο αρχείο, όπου λαμβάνονται οι τιμές των χαρακτηριστικών από τα περιεχόμενά του προς μετασχηματισμό. Φυσικά, τα χαρακτηριστικά που επιλέγονται προς εξαγωγή είναι σύμφωνα με αυτά που έχει επιλέξει ο Αναλυτής.

Διάγραμμα Ροής #2: Ο αναλυτής δεδομένων ψάχνει για δεδομένα. Τα επιλεγμένα δεδομένα φορτώνονται στον διακομιστή ανάλυσης. Απεικόνιση χωρίς αυθεντικοποίηση.

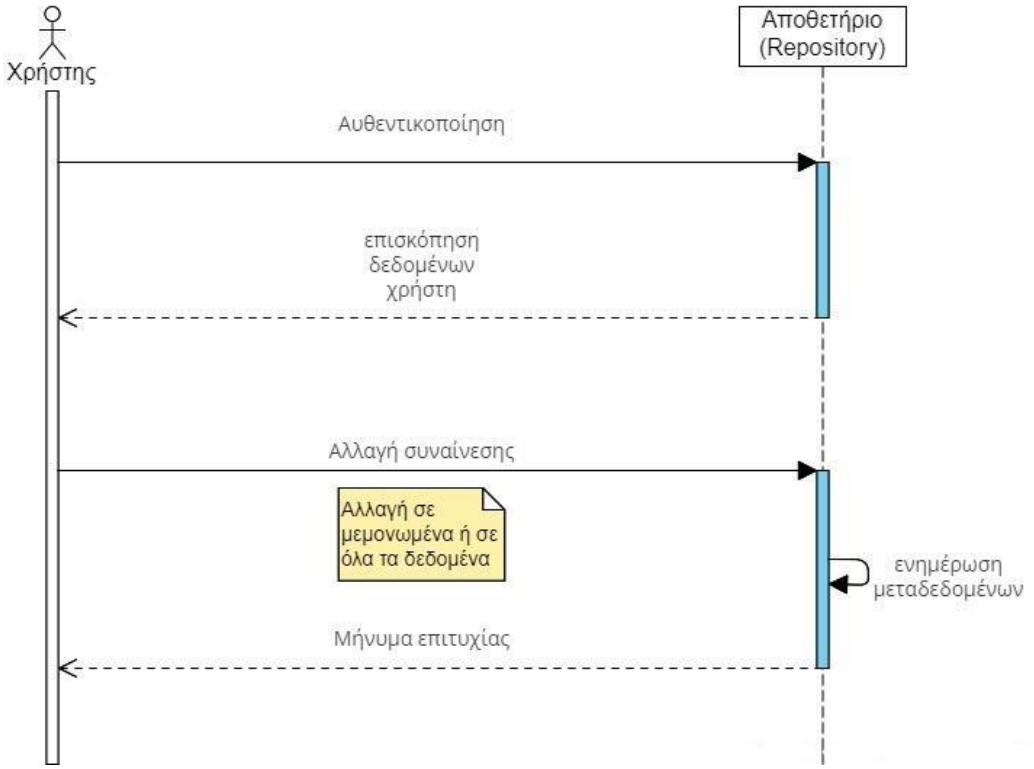


Σχήμα 3.4: Λεπτομερής απεικόνιση ακολουθίας αναζήτησης δεδομένων

Έπειτα λαμβάνει χώρα η προσάρτηση των χαρακτηριστικών στο νέο μετασχηματισμένο αρχείο. Τα δεδομένα προσαρτώνται ως νέα γραμμή, καθώς κάθε επανάληψη αντικατοπτρίζει το κάθε αρχείο. Η επανάληψη τερματίζει όταν φτάσουμε στο τελευταίο στοιχείο της λίστας. Το μέγεθος της λίστας είναι η τερματική συνθήκη (*loop guard*). Μετά το πέρας των επαναλήψεων, τα μετασχηματισμένα αρχεία αποστέλλονται στον *Φορτωτή*, ο οποίος θα τα προωθήσει στον *Διακομιστή Ανάλυσης*. Τέλος πριν την προώθηση, θα δημιουργηθούν και κάποια πρόσθετα μεταδεδομένα μορφής JSON για τον *Διακομιστή Ανάλυσης*, ώστε αυτός να γνωρίζει π.χ τον τύπο των δεδομένων (αλφαριθμητικός, δεκαδικός, ακέραιος, κλπ.). Η ακολουθία ύστερα από αυτό το βήμα συνεχίζει σύμφωνα με το Σχήμα 3.3.

Το σχήμα 3.5 αποτελεί το διάγραμμα ακολουθίας της ενημέρωσης ή ανάκλησης συναίνεσης για τα δεδομένα του χρήστη. Η ακολουθία αυτή δεν αποτελεί μέρος των τεσσάρων βασικών διεργασιών που δείξαμε στο Σχήμα 3.2, αλλά είναι μία από τις κύριες λειτουργικές απαιτήσεις που παραθέσαμε στο κεφάλαιο 2. Ο χρήστης πρέπει πρώτα να αυθεντικοποιηθεί με την πλατφόρμα, σε αυτήν την περίπτωση με το *Αποθετήριο*, όπου και θα του παρουσιαστεί μια επισκόπηση των δεδομένων που έχει ανεβάσει στην πλατφόρμα. Εδώ υπάρχουν δύο επιλογές: ο χρήστης μπορεί να αλλάξει τη συναίνεσή του για ορισμένα δεδομένα ή για όλα τα δεδομένα του μονομιάς. Η αλλαγή αυτή μπορεί να είναι ή αλλαγή π.χ. στη διαθεσιμότητα των δεδομένων ή οριστική ανάκληση. Σε οποιαδήποτε εκ των δύο περιπτώσεων, τα μεταδεδομένα ενημερώνονται από το *Αποθετήριο* σύμφωνα με τις νέες τιμές. Φυσικά η οριστική ανάκληση συνεπάγεται σε διαγραφή των δεδομένων αλλά τα μεταδεδομένα ενημερώνονται και δεν διαγράφονται εντελώς, καθώς θα περιέχουν κάποιες ελάχιστες πληροφορίες για μελλοντικές αναφορές.

Διάγραμμα Ροής #3: Ο χρήστης ανακαλεί τη συναίνεσή του.



Σχήμα 3.5: Απεικόνιση ακολουθίας ανάκλησης συναίνεσης

Τέλος το σχήμα 3.6 αποτελεί ένα βασικό σχήμα της ΒΔ που θα χρησιμοποιήσουμε ως οδηγό στο κεφάλαιο της υλοποίησης. Το σχήμα αυτό αντικατοπτρίζει την αναζήτηση και επιλογή των δεδομένων προς ανάλυση. Οι βασικοί πίνακες είναι τέσσερις:

- *User (Χρήστης)*

  Περιέχει πληροφορίες σχετικά με τον Αναλυτή. Δεν θα πρέπει να μπερδεύεται με τον *Τυπικό Χρήστη*, ο οποίος μεταφορτώνει τα δεδομένα του στην πλατφόρμα. Ο πίνακας αποθηκεύει βασικές πληροφορίες για τον Αναλυτή, όπως όνομα χρήστη, διεύθυνση ηλεκτρονικού ταχυδρομείου, κατακερματισμένο κωδικό πρόσβασης και τον οργανισμό στον οποίο ανήκει. Κύριο κλειδί του πίνακα αποτελεί το πεδίο user_id, τύπου uuid (θα εξηγηθεί στο κεφάλαιο 4). Ο πίνακας χρησιμοποιείται για αυθεντικοποίηση

- *SearchQuery (Ερώτημα Αναζήτησης)*

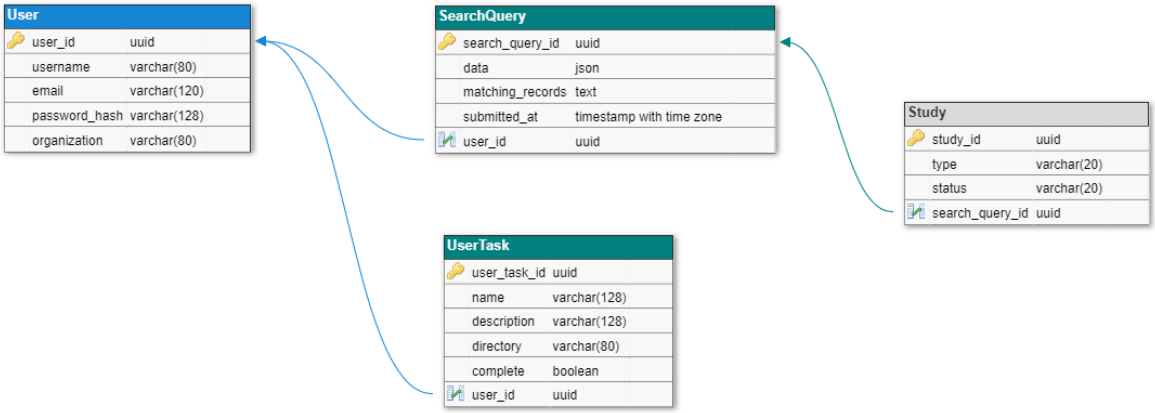
  Αποθηκεύει πληροφορίες σχετικά με το ερώτημα αναζήτησης που υπέβαλε ο αναλυτής. Κύριο κλειδί είναι το πεδίο search_query_id, επίσης τύπου uuid, ενώ αποθηκεύουμε και το αναγνωριστικό του αναλυτή ως αναφορά στον πίνακα *User* (ξένο κλειδί). Η σχέση είναι "ένα προς πολλά". Το πεδίο data θα περιέχει τα δεδομένα που αποστέλλονται από τον αναλυτή ως μορφή ερωτήματος. Τα δεδομένα αυτά αποτελούν τα φίλτρα συναίνεσης, ο χρόνος λήξης του πειράματος, τα χαρακτηριστικά προς εξαγωγή κ.ά. Ο τύπος της στήλης θα είναι τύπου JSON, καθώς μας βοηθάει να αποθηκεύσουμε πολύπλοκες δομές στην ίδια στήλη. Το πεδίο matching_records θα είναι ουσιαστικά η λίστα με τα PIDs που αναφέραμε παραπάνω. Καθώς η δομή λίστας δεν υποστηρίζεται στην SQL, ο τύπος της θα είναι αλφαριθμητικού αγνώστου μήκους *(text)*. Έτσι, μπορεί να μετατραπεί εύκολα από και σε δομή λίστας, για παράδειγμα λίστας Python. Το πεδίο submitted_at υπάρχει για λόγους προέλευσης και αποθηκεύει μια χρονική σήμανση με ζώνη ώρας για το πότε υποβλήθηκε το ερώτημα. Επίσης χρησιμεύει για τον υπολογισμό του χρόνου λήξης του ερωτήματος.

- *Study (Μελέτη)*

  Σε αυτόν τον πίνακα κρατάμε πληροφορίες για τις μελέτες που πρόκειται να υποβληθούν ή υποβλήθηκαν από τον αναλυτή. Το κύριο κλειδί είναι φυσικά το αναγνωριστικό της μελέτης *(study_id)*, ενώ το ξένο κλειδί είναι το αναγνωριστικό του ερωτήματος αναζήτησης ως αναφορά στον πίνακα *SearchQuery*. Η σχέση εδώ είναι "ένα προς ένα". Το πεδίο type περιέχει τον τύπο της μελέτης (ανάλυση, σύνθεση, ανωνυμοποίηση). Το πεδίο status αντιπροσωπεύει την κατάσταση του κάθε ερωτήματος. Για παράδειγμα ένα ερώτημα μπορεί να έχει υποβληθεί, να έχει λήξει (όταν έχει υποβληθεί και η ημερομηνία λήξης έχει περάσει) ή να είναι εκκρεμές (καθώς ο αναλυτής μπορεί να θέλει να το συνεχίσει σε αργότερη χρονική στιγμή).

- *UserTask (Εργασία Χρήστη)*

  Κρατάει πληροφορίες για τις *ουρές εργασιών* σχετικές με το χρήστη (αναλυτική εξήγηση ακολουθεί στο κεφάλαιο 4). Συνοπτικά, οι εργασίες αυτές αποθηκεύονται στην ΒΔ καθώς πρέπει να παραμένει ενεργό το πλαίσιο εκτέλεσής τους, όταν ένα αίτημα HTTP τερματίζεται, ώστε να μην χαθούν οι πληροφορίες για αυτές. Μια τέτοια εργασία αναπαρίσταται με ένα αναγνωριστικό (κύριο κλειδί), ένα όνομα, μια περιγραφή και το αναγνωριστικό του χρήστη ως ξένο κλειδί. Η σχέση είναι "ένα προς πολλά". Ακόμη αποθηκεύουμε το όνομα του καταλόγου στον οποίο πρόκειται να εγγραφούν αρχεία (αν ισχύει στην εκάστοτε περίπτωση - το πεδίο αυτό μπορεί να είναι και κενό). Το πεδίο complete είναι μια δυαδική τιμή, το οποίο υποδεικνύει εάν η εργασία είναι ακόμα σε εξέλιξη.



Σχήμα 3.6: Απεικόνιση ενός βασικού σχήματος της ΒΔ

3.4 Επίλογος

Στο παρόν κεφάλαιο δώσαμε μια γενική εικόνα των λειτουργικών αλλά και των μη λειτουργικών απαιτήσεων του συστήματος. Επίσης αναλύσαμε την αρχιτεκτονική του συστήματος, όπου αναζητήσαμε και εξηγήσαμε την επιλογή των δομικών στοιχείων αυτού. Επιπλέον απαριθμήσαμε τις διεργασίες που λαμβάνουν χώρα σε αυτό, συνοδεύοντας τες από διαγράμματα ροής ώστε να ενισχύσουμε την λογική της ιδέας μας. Στο κεφάλαιο που ακολουθεί συζητάμε την υλοποίηση του πρωτοτύπου βασιζόμενοι στις απαιτήσεις και την αρχιτεκτονική που αναλύσαμε στο παρόν κεφάλαιο. Παράλληλα παραθέτουμε τις τεχνολογίες και τους λόγους που τις χρησιμοποιήσαμε συνοδεύοντας το κείμενο και με απτά αποσπάσματα κώδικα.

Κεφάλαιο 4ο: Υλοποίηση

4.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζουμε την υλοποίηση του πρωτοτύπου, αναλύοντας τις τεχνολογίες που χρησιμοποιήθηκαν, εικόνες από την διεπαφή χρήστη καθώς και μέρη πηγαίου κώδικα.

Η κατευθυντήρια γραμμή κατά την εφαρμογή υλοποίησης του πρωτοτύπου ήταν η εξής:

i. Για το χειρισμό των δεδομένων, χρησιμοποιήθηκαν καθιερωμένες μορφές δεδομένων και πρότυπα, ώστε να αποφευχθούν παγίδες και εμπόδια που προκύπτουν κατά το σχεδιασμό από το μηδέν αλλά και να προωθηθεί η επαναχρησιμοποίηση με διάφορα άλλα συστήματα.
ii. Για τα κομμάτια του λογισμικού, επαναχρησιμοποιούμε υπάρχοντα δοκιμασμένα στοιχεία όπως για την ασφαλή αποθήκη δεδομένων, τον διακομιστή της βάσης δεδομένων και το περιβάλλον ασφαλούς ανάλυσης των δεδομένων. Έτσι η προσπάθεια επικεντρώνεται στην ενορχήστρωση και ενσωμάτωση αυτών των στοιχείων σε μια λειτουργική πλατφόρμα, η οποία εξυπηρετεί τον σκοπό της παροχής ελεγχόμενης ανάλυσης δεδομένων σεβόμενη το απόρρητο.

4.2 Υλοποίηση ασφαλούς περιβάλλοντος αποθήκευσης δεδομένων

Το περιβάλλον αποθήκευσης δεδομένων αποτελείται από το αποθετήριο δεδομένων, το οποίο έχει την μορφή ηλεκτρονικής βιβλιοθήκης, καθώς και την υπηρεσία για τις ρυθμίσεις του πειράματος από τους αναλυτές. Παρακάτω παραθέτουμε αναλυτικά την μεθοδολογία για την υλοποίηση τους, συνοδευόμενη από κομμάτια κώδικα και παραδείγματα.

4.2.1 Υλοποίηση ασφαλούς αποθήκης δεδομένων

4.2.1.1 Τεχνολογία

Για την ασφαλή αποθήκευση και ανάκτηση των δεδομένων στην πλατφόρμα θα χρησιμοποιήσουμε το Invenio framework [42], μια ανεπτυγμένη βιβλιοθήκη ανοικτού κώδικα εφαρμοσμένη σε python, η οποία χρησιμοποιείται για τη δημιουργία αποθετηρίων δεδομένων. Η βιβλιοθήκη αυτή μας επιτρέπει να αποθηκεύουμε και να οργανώνουμε σύνολα δεδομένων που μεταφορτώνονται από τις εφαρμογές των χρηστών καθώς και να διαχειριζόμαστε τον έλεγχο πρόσβασης.

Παρακάτω αναλύουμε τα χαρακτηριστικά οφέλη που προσφέρει κατά την ενσωμάτωσή του στο πρωτότυπο:

- *Ευέλικτο μοντέλο δεδομένων*

  Το πρότυπο JSON χρησιμοποιείται ως σχήμα περιγραφής των δεδομένων, το οποίο είναι εύκολο στην σύνταξη και στην αναγνωσιμότητα. Τα εισερχόμενα δεδομένα στην πλατφόρμα θα χρησιμοποιούν αυτό το πρότυπο σε μορφή μεταδεδομένων, γεγονός που θα μας διευκολύνει στην αναζήτηση και κατ' επέκταση ανάκτηση των δεδομένων.

- *Ισχυρή μηχανή αναζήτησης*

Ανεξάρτητα από το μέγεθος και το πλήθος των δεδομένων, η βιβλιοθήκη αυτή μας παρέχει γρήγορους χρόνους αναζήτησης, χρησιμοποιώντας δείκτες αντί ανάγνωσης από την βάση δεδομένων.

- *Δυνατότητα δημιουργίας εκδόσεων*

  Χρησιμοποιώντας εκδόσεις για τα δεδομένα της πλατφόρμας, αποφεύγουμε διπλότυπες εγγραφές, διένεξη πληροφοριών και καθαρό ιστορικό για τα αρχεία, διευκολύνοντας έτσι την καταγραφή ελέγχου.

- *Διεπαφές Προγραμματισμού Εφαρμογών (APIs)*

  Η βιβλιοθήκη προσφέρει εύκολα προσαρμόσιμες διεπαφές προγραμματισμού εφαρμογών, τις οποίες μπορούμε να συνδυάσουμε αλλά και να επεκτείνουμε, για να ταιριάζουν στις απαιτήσεις.

4.2.1.2 Μεταδεδομένα

4.2.1.2.1 Σχήμα βάσης δεδομένων

Όπως αναφέρθηκε παραπάνω, όταν μια εφαρμογή στέλνει δεδομένα στην πλατφόρμα, θα πρέπει να παρέχει και ένα ελάχιστο σύνολο μεταδεδομένων που θα τα περιγράφει. Τα μεταδεδομένα θεωρούνται εγγραφές τα οποία μεταφορτώνονται στην αποθήκη δεδομένων και πρέπει να συμμορφώνονται με ένα δηλωθέν σχήμα, στην περίπτωσή μας μορφής JSON. Το σχήμα αυτό διαχειρίζεται την εσωτερική επικύρωση της δομής των εγγραφών που είναι αποθηκευμένες στη βάση δεδομένων (ουσιαστικά ορίζουμε τη δομή του πίνακα στη βάση δεδομένων). Κατόπιν της μεταφόρτωσης, η εγγραφή μπορεί να συνδεθεί με ένα ή περισσότερα αρχεία δεδομένων. Μπορούμε εύκολα να καθορίσουμε το σχήμα μας, με το να επεξεργαστούμε το προεπιλεγμένο αρχείο που μας παρέχεται όταν εγκαθιστούμε την βιβλιοθήκη. Ο κώδικας 4.1 απεικονίζει το προεπιλεγμένο σχήμα των εγγραφών, όπως αυτό παρέχεται από την βιβλιοθήκη. Ο κώδικας 4.2 απεικονίζει την επέκταση του αρχείου αυτού.

Κώδικας 4.1: Προεπιλεγμένο σχήμα εγγραφής μορφής JSON

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://localhost/schemas/records/record-v1.0.0.json",
  "additionalProperties": false,
  "title": "Schema v1.0.0",
  "type": "object",
  "properties": {
    "$schema": {
      "description": "JSONSchema for the record",
      "type": "string"
    },
    "_bucket": {
      "description": "UUID of the deposit bucket.",
      "type": "string"
    },
    "_files": {
```

```
      "description": "Describe information needed for files in records.",
      "type": "array",
      "items": {
        "description": "Describes the information of a single file in the
record.",
        "properties": {
          "key": {
            "description": "Key (filename) of the file.",
            "type": "string"
          },
          "file_id": {
            "description": "UUID of the FileInstance object.",
            "type": "string"
          },
          "bucket": {
            "description": "UUID of the bucket to which this file is assigned.",
            "type": "string"
          },
          "checksum": {
            "description": "Checksum the file. Starts with '<algorithm>:'
prefix, e.g.: 'md5:1234abcd...'",
            "type": "string"
          },
          "size": {
            "description": "Size of the file in bytes.",
            "type": "integer"
          },
          "version_id": {
            "description": "UUID of the ObjectVersion object.",
            "type": "string"
          }
        }
      }
    },
    "id": {
      "description": "Record identifier.",
      "type": "string"
    },

}
```

Τα πεδία των δομών JSON μπορούν να είναι βασικοί τύποι δεδομένων (λ.χ. αλφαριθμητικοί, ακέραιοι κτλ.) αλλά και σύνθετοι όπως αντικείμενα ή πίνακες αντικειμένων. Παραπάνω απεικονίζεται η δομή μιας εγγραφής, όπου τα αντίστοιχα πεδία συμπληρώνονται αυτόματα κατά τη δημιουργία της. Η προσοχή μας εστιάζεται στο πεδίο *"id"*, το οποίο είναι και το μοναδικό αναγνωριστικό της εγγραφής. Αυτό το πεδίο ονομάζεται και *Persistent Identifier*. Βάσει αυτού θα κινηθεί η αναζήτηση και η εξαγωγή των δεδομένων για ανάλυση.

Όπως αναφέρθηκε και στο κεφάλαιο 3, τα μεταδεδομένα θα πρέπει να υποδεικνύουν την ύπαρξη χαρακτηριστικών στα σύνολα δεδομένων αλλά θα αντικατοπτρίζουν και την συγκατάθεση του χρήστη. Παρακάτω επεκτείνουμε το αρχείο, ώστε να τα αναπαριστά.

Η συγκατάθεση του χρήστη αναπαρίσταται ως πίνακας αντικειμένων. Εάν ο χρήστης δεν δώσει μια ενιαία συγκατάθεση για όλα τα δεδομένα που του ανήκουν, μπορεί να δώσει διαφορετικές συγκαταθέσεις για μεμονωμένα δεδομένα ή κατηγορίες αυτών. Σε αυτήν την περίπτωση θα έχουμε περισσότερα από ένα αντικείμενα συγκατάθεσης. Για αυτό το λόγο θέτουμε το πεδίο "*data_type*" (τύπος δεδομένων), το οποίο είναι ένας πίνακας αλφαριθμητικών τιμών και αντικατοπτρίζει το σύνολο των δεδομένων, στο οποίο θα εφαρμοστεί η εκάστοτε συγκατάθεση.

Εάν ο χρήστης επιθυμεί να δώσει συγκατάθεση:

- για κατηγορίες δεδομένων, τότε ο πίνακας θα περιέχει τουλάχιστον μία τιμή από αυτές, π.χ. "Δημογραφικά".
- για συγκεκριμένα δεδομένα, τότε ο πίνακας θα περιέχει τουλάχιστον μια τιμή από αυτά, π.χ. "Ηλικία".
- για όλα τα δεδομένα, τότε ο πίνακας θα περιέχει μία ακριβώς τιμή, η οποία θα είναι "όλα τα δεδομένα".

Φυσικά μπορεί να υπάρξει και συνδυασμός κατηγοριών με μεμονωμένα δεδομένα, όπου ο πίνακας θα περιέχει ονόματα κατηγοριών και ονόματα δεδομένων.

Το επόμενο πεδίο που παρατηρούμε είναι το "*until*" (μέχρι), το οποίο υποδεικνύει μέχρι πότε θα ισχύει η συναίνεση στα δεδομένα του χρήστη. Θα έχει μορφή ημερομηνίας, με μοτίβο EEEE-MM-HH. Κατόπιν της δηλωθείσας ημερομηνίας, η συναίνεση στα δεδομένα λήγει και δεν είναι πλέον προσβάσιμα για πειράματα, εκτός και αν ο χρήστης ανανεώσει την ημερομηνία.

Το πεδίο "processing" (επεξεργασία) είναι επίσης πίνακας αλφαριθμητικών και δηλώνει την πρόθεση του χρήστη για τους τύπους ανάλυσης που θα επιτρέπεται να υποστούν τα δεδομένα. Παραδείγματα αποτελούν οι τιμές "ανάλυση" και "μελέτη προφίλ".

Προχωρώντας, το πεδίο "purpose" (σκοπός) απαρτίζεται από πολλαπλές τιμές αλφαριθμητικών και δηλώνει υπό ποιο σκοπό μπορεί να γίνει η πρόσβαση των δεδομένων. Παραδείγματα αποτελούν "ακαδημαϊκή έρευνα" και "εμπορική έρευνα".

Τελευταίο πεδίο της συγκατάθεσης αποτελεί ο "recipient" (παραλήπτης), το οποίο συμπληρώνεται αυτόματα από την εφαρμογή του χρήστη, σύμφωνα με την εταιρία/οργανισμό στην οποία ανήκει. Για να προσπελαστούν τα δεδομένα σε διεξαγόμενη έρευνα, ο αναλυτής πρέπει να ανήκει στην ίδια εταιρία/οργανισμό, εκτός και αν ο χρήστης έχει δώσει συγκατάθεση για συνδυαζόμενη επεξεργασία.

Εκτός της συγκατάθεσης, ορίζουμε ακολούθως το πεδίο "data_summary" (σύνοψη δεδομένων) ως έναν πίνακα αλφαριθμητικών. Το πεδίο αυτό ορίζεται μόνο για ανάγνωση, δηλαδή οι εφαρμογές των χρηστών δεν θα μπορούν να δημιουργήσουν ή να αλλάξουν τιμές σε αυτό. Συνεπώς οι εισερχόμενες τιμές θα εξαρτώνται από τα περιεχόμενα του αρχείου.

Τέλος, έχουμε το πεδίο "patient_id" (αναγνωριστικό ασθενή), το οποίο είναι επίσης μόνο για ανάγνωση και συμπληρώνεται αυτόματα, σύμφωνα με το εισερχόμενο αρχείο. Ο ρόλος του είναι βοηθητικός και

συμβάλλει στην παρουσίαση των δεδομένων, όταν αυτά ανακτώνται και εσωτερικά προς αναφορά, όταν λαμβάνει χώρα η εξαγωγή. Επιπρόσθετα τα στοιχεία θα πρέπει να είναι μοναδικά και το μέγεθος του πίνακα να είναι τουλάχιστον ένα, για ευνόητους λόγους.

Κώδικας 4.2: Προσαρμοσμένο σχήμα εγγραφής μορφής JSON

```
{
  ...
  "properties": {
    ...
    "id": {
      "description": "Invenio record identifier.",
      "type": "string"
    },
    "consent": {
      "description": "Type of consent given.",
      "type": "array",
      "items": {
        "title": "",
        "type": "object",
        "properties": {
          "data_type": {
            "type": "array",
            "items": {
              "type": "string",
              "enum": ["HealthRecord", "Demographic",
"PhysicalCharacteristic", "Language", "Country"]
            },
            "minItems": 1,
            "uniqueItems": true
          },
          "until": {
            "description": "Indicates the time interval, after which the
type of consent given on the data expires.",
            "type": "string",
            "format": "date"
          },
          "processing": {
            "description": "Indicates type of processing for the data.",
            "type": "array",
            "items": {
              "type": "string",
              "enum": ["DiscloseByTransmission", "Disseminate",
"MakeAvailable", "Transmit", "Share", "Adapt", "Align", "Anonymise",
"Combine", "Derive", "PseudoAnonymise", "Restrict", "Analyse", "Consult",
"Profiling", "Retrieve"]
```

```
            },
            "minItems": 1,
            "uniqueItems": true
          },
          "purpose": {
            "description": "Indicates according to which purpose the data
can be accessed",
            "type": "array",
            "items": {
              "type": "string",
              "enum": ["SellDataToThirdParties", "SellInsightsFromData",
"SellProductsToDataSubject", "SellTargetedAdvertisements",
"AcademicResearch", "CommercialResearch", "NonCommercialResearch",
"AccessControl", "FraudPreventionAndDetection", "IdentifyVerification",
"ServiceOptimization", "ServicePersonalization", "ServiceProvision"]
            },
            "minItems": 1,
            "uniqueItems": true
          },
          "recipient": {
            "description": "Indicates the name of the company to which
the data belong to",
            "type": "string",
            "enum": ["InstitutionA","InstitutionB","CompanyA","CompanyB"]
          }
        },
        "additionalProperties": false,
        "required": ["data_type", "purpose", "until", "processing",
"recipient"]
      }
    },
    "data_summary": {
      "title": "The Data Summary Schema",
      "description": "Indicates if specific data exist in the file
object(s)",
      "type": "array",
      "$comment": "List that indicates which processable attributes are
present in each file",
      "items": {
        "type": "string"
      },
      "uniqueItems": true,
      "minItems": 1
    },
    "patient_id": {
      "description": "Patient Identifier.",
```

```
      "type": "string"
    }
  },
  "required": ["$schema", "id", "consent","data_summary", "patient_id"]
}
```

4.2.1.2.2 Επικύρωση και παρουσίαση δεδομένων

Ένα από τα πακέτα που χρησιμοποιεί η βιβλιοθήκη είναι το marshmallow. Το πακέτο αυτό συμβάλλει στην επικύρωση των εισερχόμενων δεδομένων. Ακόμη πραγματοποιεί deserialization των δεδομένων σε αντικείμενα επιπέδου εφαρμογής και αντίστοιχα serialization στα αντικείμενα σε βασικούς τύπους Python. Τα σειριακά αντικείμενα μπορούν στη συνέχεια να αναπαρασταθούν σε τυπικές μορφές όπως JSON στην περίπτωση μας, για να τα χρησιμοποιήσουμε σε ένα HTTP API. Με άλλα λόγια το πακέτο είναι υπεύθυνο για τον μετασχηματισμό των δεδομένων από και σε Python, κατά την είσοδο ή την έξοδο τους αντίστοιχα από την εφαρμογή. Τα σχήματα βασίζονται σε κλάσεις, συνεπώς προωθείται η επαναχρησιμοποίηση κώδικα και η εύκολη διαμόρφωση. Ο κώδικας 4.3 παρουσιάζει το σχήμα κλάσεων για τις εγγραφές που πραγματοποιεί όλα τα παραπάνω. Πρέπει να υπογραμμίσουμε, ότι τα σχήματα κλάσεων πρέπει να ακολουθούν τις ιδιότητες που έχουμε ορίσει στο JSON αρχείο, ειδάλλως το API δεν θα δέχεται ή επιστρέφει τις δηλωθείσες τιμές.

Κώδικας 4.3: Σχήμα κλάσεων για επικύρωση δεδομένων

```
from invenio_records_rest.schemas import Nested, StrictKeysMixin
from invenio_records_rest.schemas.fields import DateString, GenFunction,
PersistentIdentifier, SanitizedUnicode
from marshmallow import fields, missing, validate

class ConsentSchemaV1(StrictKeysMixin):
    """Schema for the user consent."""
    data_type = fields.List(SanitizedUnicode(), required=True,
validate=validate.Length(min=1))
    until = DateString(required=True)
    processing = fields.List(SanitizedUnicode(), required=True,
validate=validate.Length(min=1))
    purpose = fields.List(SanitizedUnicode(), required=True,
validate=validate.Length(min=1))
    recipient = SanitizedUnicode(required=True)

class MetadataSchemaV1(StrictKeysMixin):
    """Schema for the record metadata."""
    id = PersistentIdentifier()
    consent = fields.Nested(ConsentSchemaV1, many=True, required=True)
    data_summary = fields.List(SanitizedUnicode(), required=True,
validate=validate.Length(min=1))
    patient_id = SanitizedUnicode(required=True)
    _schema = GenFunction(...)
```

```
class RecordSchemaV1(StrictKeysMixin):
    """Record schema."""
    metadata = fields.Nested(MetadataSchemaV1)
    created = fields.Str(dump_only=True)
    revision = fields.Integer(dump_only=True)
    updated = fields.Str(dump_only=True)
    links = fields.Dict(dump_only=True)
    id = PersistentIdentifier()
    files = GenFunction(...)
```

Κάθε δηλωθέν πεδίο πρέπει να ανατεθεί σε μία κλάση υπό το πακέτο fields (πεδία) του marshmallow, η οποία καθορίζει και τον τύπο τους. Οι κλάσεις αυτές είναι φυσικά οι βασικοί τύποι, όπως ακέραιοι, δεκαδικοί αλλά και πιο σύνθετοι όπως εμφωλευμένοι, λεξικά, λίστες κτλ.

Η δήλωση των πεδίων ξεκινάει με την κλάση RecordSchemaV1, η οποία περιέχει τα βασικά πεδία μιας εγγραφής: τα μεταδεδομένα (metadata), τη χρονική σήμανση δημιουργίας (created) και τελευταίας τροποποίησης (updated), το αναγνωριστικό έκδοσης (revision) και τους υπερσυνδέσμους που συνδέονται με τον εκάστοτε πόρο. Η βιβλιοθήκη μας προσφέρει και την υλοποιημένη κλάση PersistentIdentifier, με την οποία το πεδίο θα έχει έναν αριθμό ως μοναδικό αναγνωριστικό, το οποίο παραμένει αμετάβλητο με την πάροδο του χρόνου, αποφεύγοντας έτσι τις “σπασμένες αναφορές”. Με τον όρο αυτό εννοούμε εγγραφές, οι οποίες έχουν ως αναγνωριστικό που έχει διαγραφεί ή έχει τροποποιηθεί στο σύστημα, γεγονός που μπορεί να προκαλέσει προβλήματα κατά την ανάκτηση δεδομένων. Τέλος έχουμε τα αρχεία, τα οποία ορίζονται ως αντικείμενο γενικοποιημένης συνάρτησης, μια επίσης υλοποιημένη κλάση, η οποία υποχρεώνει τα πεδία της συνάρτησης να υφίστανται πάντα deserialization.

Οι 3 συναρτήσεις είναι βοηθητικές και χρησιμοποιούνται για την ανάκτηση buckets, αρχείων και σχημάτων αντίστοιχα, σύμφωνα με το δοθέν context. Ακόμη οι κλάσεις επεκτείνουν την κλάση StrictKeysMixin, η οποία διασφαλίζει ότι μόνο συγκεκριμένες τιμές είναι αποδεκτές στο εκάστοτε πεδίο. Οι τιμές αυτές πρέπει να δηλωθούν στο σχήμα JSON υπό το πεδίο enum ως λίστα αλφαριθμητικών. Ακόμη η βιβλιοθήκη Invenio μας παρέχει τον πρόσθετο τύπο δεδομένων SanitizedUnicode για αλφαριθμητικές τιμές (φυσικά είναι υπο-κλάση της κλάσης String), η οποία διορθώνει τυχόν προβληματικούς χαρακτήρες Unicode, λ.χ. το “\u200b” (κενός χαρακτήρας μηδενικού μήκους).

Για να αναπαραστήσουμε σύνθετα πεδία όπως τα μεταδεδομένα και τη συναίνεση, πρέπει να χρησιμοποιούμε μια κλάση και να περάσουμε το όνομά της ως παράμετρο στον δομητή Nested. Κατόπιν δηλώνουμε τα υπο-πεδία ως ιδιότητες της με τον κατάλληλο δομητή-τύπο, έως ότου δεν υπάρχει κάποια άλλη σύνθετη δομή. Στον παραπάνω κώδικα, συμπληρώνουμε τα πεδία κάθε σύνθετης κλάσης με τον κατάλληλο τύπο, ώστε να συνάδουν με τους τύπους του JSON αρχείου που συμπληρώσαμε στον κώδικα 4.2.

Έχουμε φυσικά τη δυνατότητα να περάσουμε παραμέτρους σε κάθε τύπο, οι οποίοι θα καθορίζουν τις ιδιότητες των πεδίων. Περνάμε την παράμετρο dump_only ως αληθή όταν το αντίστοιχο πεδίο πρέπει

να είναι μόνο για ανάγνωση (μπορεί να μεταβληθεί μόνο εσωτερικά από την εφαρμογή). Ομοίως θέτουμε required ως αληθές όταν το πεδίο πρέπει να είναι πάντοτε παρόν κατά την είσοδο των δεδομένων στο σύστημα. Η παράμετρος many υποδηλώνει ότι το εκάστοτε πεδίο θα μπορεί να έχει ένα ή περισσότερα αντικείμενα της δηλωμένης κλάσης. Ακόμη χρησιμοποιούμε λειτουργικότητα επικύρωσης για το μήκος των τιμών, θέτοντας το Length(min=1), όπου κενές τιμές δεν θα είναι αποδεκτές.

4.2.1.2.3 Σχήμα Αναζήτησης Δεδομένων

Όπως προαναφέρθηκε, η βιβλιοθήκη Invenio αποφεύγει την ανάγνωση δεδομένων απευθείας από τη βάση δεδομένων και αντί αυτού δημιουργεί και χρησιμοποιεί ευρετήρια μηχανής αναζήτησης για την ανάκτηση των δεδομένων. Τα ευρετήρια αυτά δημιουργούνται με την βοήθεια της Elasticsearch, μια ισχυρή μηχανή αναζήτησης ανοικτού κώδικα, που παρέχει πλήθος αποτελεσμάτων με ελάχιστο χρόνο καθυστέρησης. Ένα ευρετήριο είναι μια συλλογή εγγράφων (τα μεταδεδομένα που συνοδεύουν κάθε αρχείο στην περίπτωσή μας), που μοιράζονται κοινά χαρακτηριστικά. Είναι η οντότητα υψηλότερου επιπέδου στην οποία υποβάλλουμε ερωτήματα. Ουσιαστικά θεωρείται παρόμοιο με μια βάση δεδομένων σε ένα σχεσιακό σχήμα βάσης δεδομένων. Κάθε ευρετήριο προσδιορίζεται από ένα όνομα που χρησιμοποιείται για να αναφέρεται σε αυτό, καθώς και για λειτουργίες αναζήτησης, ενημέρωσης και διαγραφής ενάντια στα έγγραφά του. Για να καταστήσουμε τα δεδομένα διαθέσιμα για αναζήτηση, χρησιμοποιούμε ένα σχήμα JSON, παρόμοιο με των εικόνων 5.2 και 5.3. Βάσει αυτού του σχήματος θα δημιουργηθεί το αντίστοιχο ευρετήριο όπου θα δημιουργηθούν και τα πεδία με τον αντίστοιχο τύπο. Ο κώδικας 4.4 απεικονίζει ένα απλό σχήμα του ευρετηρίου των εγγραφών.

Κώδικας 4.4: Σχήμα ευρετηρίου εγγραφών

```
{
  "mappings": {
    "date_detection": false,
    "numeric_detection": false,
    "properties": {
      "$schema": {
        "type": "text",
        "index": false
      },
      "title": {
        "type": "text",
        "copy_to": "suggest_title"
      },
      "suggest_title": {
        "type": "completion"
      },
      "id": {
        "type": "keyword"
      },
      "consent": {
        "type": "object",
```

```
        "properties": {
          "data_type": {
            "type": "text"
          },
          "until": {
            "type": "date",
            "format": "date"
          },
          "processing": {
            "type": "text"
          },
          "purpose": {
            "type": "text"
          },
          "recipient": {
            "type": "text"
          }
        }
      },
      "data_summary": {
        "type": "keyword"
      },
      "patient_id": {
        "type": "keyword"
      },
      "_created": {
        "type": "date"
      },
      "_updated": {
        "type": "date"
      }
    }
  }
}
```

Η βασική διαφορά είναι ότι όλα τα πεδία δηλώνονται ως ιδιότητες του πεδίου *mappings* (συσχετίσεις), ενώ δεν δηλώνουμε τύπους πινάκων. Επιπρόσθετα οι σύνθετες ιδιότητες δηλώνονται κανονικά ως *object*, χωρίς όμως να δηλώνουμε τους πίνακες . Για όλα τα πεδία αρκεί η συσχέτιση ονόματος με τύπο. Αυτό μας οδηγεί και στην δεύτερη διαφορά, όπου οι τύποι των εγγράφων του ευρετηρίου έχουν διαφορετικές προδιαγραφές για τη δήλωση τύπων δεδομένων. Αρχικά οι αλφαριθμητικές τιμές θέτονται ως *text* (*κείμενο*) και *keyword* (λέξη-κλειδί). Η κύρια διαφορά μεταξύ του τύπου δεδομένων κειμένου και του τύπου δεδομένων λέξεων-κλειδιών είναι ότι τα πεδία κειμένου αναλύονται κατά τη στιγμή της ευρετηρίασης ενώ τα πεδία λέξεων-κλειδιών όχι. Αυτό σημαίνει ότι, τα πεδία κειμένου αναλύονται στους μεμονωμένους όρους τους κατά την ευρετηρίαση για να επιτρέπεται η μερική αντιστοίχιση, ενώ τα πεδία λέξεων-κλειδιών ευρετηριάζονται ως έχουν και οι τιμές των φίλτρων αναζήτησης πρέπει να ταιριάζουν ακριβώς με τις τιμές των αντίστοιχων πεδίων. Ο τύπος *date* είναι φυσικά για ημερομηνίες.

Τέλος τοποθετούμε το πεδίο "*index*" ως ψευδές, όταν δεν θέλουμε το πεδίο να εισαχθεί στο ευρετήριο, συνεπώς δεν θα είναι αναζητήσιμο.

4.2.1.3 Αρχεία

Τα αρχεία των τυπικών χρηστών θα αποθηκεύονται στο αποθετήριο με την αρχική τους μορφή. Κάθε εισερχόμενο αρχείο πρέπει βέβαια να συνοδεύεται και από τα αντίστοιχα μεταδεδομένα, στα οποία θα επισυνάπτεται ως αναφορά. Για την εύρεση και ανάκτηση του αρχείου για επεξεργασία εσωτερικά της πλατφόρμας, πρέπει πρώτα να ανακτηθεί η αναφορά των μεταδεδομένων του και κατόπιν το ζητηθέν αρχείο. Υπενθυμίζουμε ότι τα ονόματα των αρχείων ανήκουν επίσης στο index των εγγραφών, γεγονός που μας επιτρέπει στη γρήγορη ανάκτηση πληροφοριών για αυτά π.χ. για αποτελέσματα πλήθους αναζήτησης.

4.2.1.4 Βασικές Διαδρομές ΔΠΕ

Η βιβλιοθήκη παρέχει μερικές βασικές διαδρομές, τις οποίες μπορούμε να επεξεργαστούμε εύκολα για τις ανάγκες της πλατφόρμας. Παρακάτω παραθέτουμε έξι, από τις οποίες θα χρησιμοποιήσουμε συχνά στην εφαρμογή, για δημιουργία/ανάκτηση μεταδεδομένων και αρχείων.

1. GET *api/records*

   Η βασική διαδρομή για την ανάκτηση όλης της συλλογής εγγραφών/μεταδεδομένων. Τα αποτελέσματα που επιστρέφει είναι ουσιαστικά ερώτημα προς την υπηρεσία Elasticsearch, οπότε και έχει τη μορφή σελιδοποιημένων αποτελεσμάτων (με προκαθορισμένη τιμή 10 αποτελέσματα ανά σελίδα)

2. POST *api/records*

   Η ίδια διαδρομή αλλά με την μέθοδο POST, δημιουργεί την εγγραφή στην συλλογή με τα παρεχόμενα δεδομένα. Φυσικά μέσω αυτής της κλήσης, τα δεδομένα προσαρτώνται στο ευρετήριο της Elasticsearch για μελλοντική ανάκτηση. Τέλος, δημιουργείται και ένας Persistent Identifier για την εγγραφή, ο οποίος είναι και μοναδικός.

3. GET *api/records/<recid>*

   Με τη διαδρομή αυτή, ανακτούμε τα μεταδεδομένα μια συγκεκριμένης εγγραφής, αλλά και τις αναφορές (links) για τα συνδεδεμένα αρχεία με αυτήν. Η διαδρομή είναι μεταβλητή λόγω του πεδίου <recid>, όπου και θα πρέπει να παρασχεθεί το αντίστοιχο αναγνωριστικό της επιθυμητής εγγραφής προς ανάκτηση.

4. PUT *api/records/<recid>*

   Ίδια με την διαδρομή 3, αλλά χρησιμοποιώντας την μέθοδο PUT στη συγκεκριμένη εγγραφή, μπορούμε να τροποποιήσουμε κάποια πεδία της.

5. PUT *api/records/<recid>/files/<filename>*

Πραγματοποιεί επισύναψη του αρχείου με το δοθέν όνομα, στην αντίστοιχη εγγραφή μεταδεδομένων. Συνήθως χρησιμοποιείται σε συνδυασμό με την διαδρομή 2.

6. GET *api/records/<recid>/files/<filename>*

   Ανακτά τα περιεχόμενα του αρχείου, σύμφωνα με το αναγνωριστικό της εγγραφής και το πλήρες όνομα του αρχείου.

4.2.2 Υλοποίηση υπηρεσίας ρυθμίσεων πειραμάτων

Η Υπηρεσία Ρυθμίσεων Πειραμάτων δημιουργήθηκε από την αρχή (*from scratch*) ακολουθώντας τις βέλτιστες πρακτικές των [43-45].

4.2.2.1 Τεχνολογία

Για την υλοποίηση της υπηρεσίας για χρήση πειραμάτων από τους αναλυτές, θα χρησιμοποιήσουμε το Flask [46], ένα Python framework για την δημιουργία εφαρμογών ιστού, το οποίο παρέχει εργαλεία, βιβλιοθήκες και διάφορες τεχνολογίες. Το Flask ταξινομείται ως ένα microframework που σημαίνει ότι αφήνει περισσότερη ελευθερία στο πώς γράφουμε και οργανώνουμε τον κώδικα μας καθώς και ότι έχει ελάχιστες έως καθόλου εξαρτήσεις από εξωτερικές βιβλιοθήκες. Κάποια από τα βασικά πλεονεκτήματά του είναι η ευελιξία του, ο ενσωματωμένος διακομιστής για development και ο γρήγορος αποσφαλματωτής (*debugger*). Επίσης παρέχει ενσωματωμένη υποστήριξη για unit testing και cookies, λειτουργία χειρισμού αιτημάτων HTTP και εύκολο deployment σε λειτουργία παραγωγής (production). Τέλος οι παρεχόμενες ΔΠΕ είναι συνεκτικές και όμορφα διαμορφωμένες για εύκολη χρήση.

Η δημιουργία μιας εφαρμογής δημιουργείται με ένα αντικείμενο τύπου *Flask* και κατόπιν, κλήσης της μεθόδου *run()* στο αντικείμενο αυτό. Στον κώδικα 4.5 παραθέτουμε μια απλή εφαρμογή. Όπως παρατηρούμε, η δημιουργία της εφαρμογής είναι αρκετά απλή. Πρώτα εισαγάγαμε την κλάση Flask. Η εφαρμογή Διασύνδεσης Πύλης Διακομιστή Ιστού θα είναι ένα αντικείμενο αυτής της κλάσης. Η ΔΠΔΙ (WSGI), είναι ένας απλός κανόνας κλήσης για διακομιστές Ιστού, για την προώθηση αιτημάτων σε εφαρμογές Ιστού ή framework γραμμένα στη γλώσσα προγραμματισμού Python.

Κώδικας 4.5: Απλή εφαρμογή Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
      return '<p>Hello world</p>'

if __name__ == '__main__':
      app.run(debug=True, port=5000)
```

Στη συνέχεια δημιουργούμε ένα instance αυτής της κλάσης. Η πρώτη παράμετρος είναι το όνομα του module ή του πακέτου της εφαρμογής. Το __name__ είναι μια βολική συντόμευση, κατάλληλη για τις περισσότερες περιπτώσεις. Είναι απαραίτητη ώστε το Flask να γνωρίζει πού να αναζητήσει πόρους όπως templates και στατικά αρχεία. Στη συνέχεια χρησιμοποιούμε τον διακοσμητή route() για να πούμε

στο Flask ποιος Ενιαίος Εντοπιστής Πόρων (*URL*) θα πρέπει να ενεργοποιεί τη συνάρτησή μας. Η συνάρτηση επιστρέφει το μήνυμα που θέλουμε να εμφανίσουμε στο πρόγραμμα περιήγησης του χρήστη. Ο προεπιλεγμένος τύπος περιεχομένου είναι HTML, επομένως το πρόγραμμα περιήγησης θα αποδώσει την HTML που συμπεριλαμβάνεται στη συμβολοσειρά.

Η συνθήκη της εντολής if είναι ο κώδικας *boilerplate* που προστατεύει τους χρήστες από την κατά λάθος κλήση του σεναρίου (script) όταν δεν υπάρχει τέτοιος σκοπός. Ο *φύλακας if* δεν θα πρέπει να παραλείπεται καθώς εάν το script εισαχθεί από κάποιο άλλο, θα ενεργοποιήσει την εκτέλεση του πρώτου κατά τον χρόνο εισαγωγής και χρησιμοποιώντας τα ορίσματα γραμμής εντολών του δεύτερου, όπου και η εκτέλεση θα είναι προβληματική. Αναλυτικότερα, πριν από την εκτέλεση του κώδικα, ο διερμηνέας της Python διαβάζει το αρχείο προέλευσης και ορίζει κάποιες ειδικές μεταβλητές/καθολικές μεταβλητές, οι οποίες ονομάζονται *μαγικές μεταβλητές (magic variables)*. Αυτές οι μεταβλητές έχουν 2 κάτω παύλες στην αρχή και στο τέλος του ονόματός τους. Εάν ο διερμηνέας της Python εκτελέσει αυτό το module (το αρχείο προέλευσης) ως κύριο πρόγραμμα, ορίζει την ειδική μεταβλητή __name__ να έχει μια τιμή "__main__". Εάν αυτό το αρχείο εισάγεται από άλλο module, το __name__ θα οριστεί στο όνομα του module που το εισάγει. Το όνομα της μονάδας είναι διαθέσιμο ως τιμή στην καθολική μεταβλητή __name__.

Για να τρέξουμε την εφαρμογή θα εκτελέσουμε: python3 <όνομα_αρχείου>.py, όπου και ο ενσωματωμένος διακομιστής που προσφέρεται θα ενεργοποιηθεί. Αυτό φυσικά προϋποθέτει την εγκατάσταση του πακέτου στο σύστημά μας (π.χ *pip install flask*). Ο διακομιστής αυτός δεν μπορεί να χειριστεί περισσότερα από ένα αιτήματα HTTP τη φορά από προεπιλογή, ενώ παρουσιάζει και θέματα scalability, συνεπώς και δεν πρέπει να χρησιμοποιηθεί σε περιβάλλον production. Από την έκδοση 0.11 και έπειτα, μπορούμε επίσης να χρησιμοποιήσουμε την εντολή *flask run* για την εκκίνηση του τοπικού διακομιστή. Με αυτόν τον τρόπο ωστόσο, πρέπει να ενημερώσουμε το Flask πώς να εισάγει την εφαρμογή. Αυτό επιτυγχάνεται ορίζοντας την τιμή της μεταβλητής περιβάλλοντος *FLASK_APP* στο αρχείο που περιέχει τον κώδικα εκκίνησης.

4.2.2.2 Βασική Δομή εφαρμογής

Όπως αναφέραμε, το Flask είναι πολύ ευέλικτο, οπότε η δομή μιας εφαρμογής εξαρτάται καθαρά από τον προγραμματιστή. Η παρούσα υποενότητα υπηρετεί ως οδηγός για το μοτίβο δομής που ακολουθήσαμε. Η δομή αυτή θα χρησιμοποιηθεί ως αναφορά παρακάτω, καθώς θα αναλύουμε λειτουργικότητες ή θα προσθέτουμε περισσότερα αρχεία. Το σχήμα 4.1 απεικονίζει την βασική δομή των καταλόγων και αρχείων.

Στον κατάλογο web τοποθετούμε ουσιαστικά όλη την λειτουργικότητα της εφαρμογής. Μέλη του είναι:

- Ένα κρυφό αρχείο .env. Περιέχει τις παραμέτρους διαμόρφωσης που χρησιμοποιούνται από την εφαρμογή. Αναλύεται περαιτέρω στην υποενότητα 4.2.2.3.
- Ένα αρχείο python config.py. Χειρίζεται την ανάγνωση των μεταβλητών διαμόρφωσης από το περιβάλλον του ΛΣ (ουσιαστικά του παραπάνω αρχείου .env). Περισσότερες λεπτομέρειες υπάρχουν επίσης στην υποενότητα 4.2.2.3.
- Ένα απλό αρχείο requirements.txt. Είναι ένας τύπος αρχείου που αποθηκεύει πληροφορίες σχετικά με όλες τις βιβλιοθήκες, τις λειτουργικές μονάδες και τα πακέτα που χρησιμοποιούνται κατά την ανάπτυξη του project. Μπορεί επίσης να αποθηκεύει όλα τα αρχεία και τα πακέτα από τα οποία εξαρτάται ή απαιτούνται για να εκτελεστεί το project.

- Ένα python αρχείο version.py. Περιέχει πληροφορίες σχετικά με την έκδοση του συστήματος, κυρίως για λόγους προέλευσης.
- Ένα python αρχείο wsgi.py. Το αρχείο αυτό περιέχει την αρχικοποίηση της εφαρμογής μας και θα καλείται από κάποιο script ή χειροκίνητα από τη γραμμή εντολών ώστε αυτή να εκκινηθεί.
- Τον κατάλογο app. Σε αυτόν περιέχονται οι διαδρομές της διεπαφής της εφαρμογής, οι διαδρομές του API, οι σελίδες HTML και τα στατικά αρχεία
- Οι κατάλογοι logs, tests και venv. Περιέχουν τα αρχεία καταγραφής, αυτόματα τεστ για την εφαρμογή και τις βιβλιοθήκες από τις οποίες εξαρτάται οι εφαρμογή αντίστοιχα.

```
web/
  app/
    api/
    __init__.py
    routes.py
    static/
    templates/
  logs/
  tests/
  venv/
  .env
  config.py
  requirements.txt
  version.py
  wsgi.py
```

Σχήμα 4.1: Βασική δομή εφαρμογής

4.2.2.3 Ρύθμιση εφαρμογής

Η τρέχουσα υποενότητα παρουσιάζει τους τρόπους δήλωσης μεταβλητών διαμόρφωσης για την εφαρμογή. Τα αρχεία διαμόρφωσης (κοινώς γνωστά ως config ή configuration files) είναι αρχεία που χρησιμοποιούνται για τη διαμόρφωση παραμέτρων και αρχικών ρυθμίσεων για τον διακομιστή.

Κώδικας 4.6: Παράδειγμα απλής διαμόρφωσης παραμέτρων Flask

```
import os
from flask import Flask
app = Flask(__name__)
app.config['SECRET_KEY'] = 'my-secret'
app.config['DATABASE'] = os.path.join(app.instance_path, 'app_db.db')
```

Ο κώδικας 4.6 παρουσιάζει ένα παράδειγμα με 2 μεταβλητές διαμόρφωσης. Το Flask μας παρέχει την ιδιότητα *config* του δημιουργηθέντος αντικειμένου, το οποίο είναι ένα λεξικό (*dictionary*), όπου και αποθηκεύουμε όλες τις παραμέτρους σχετικές με το διακομιστή. Αργότερα, μπορούμε να το χρησιμοποιήσουμε σε σημεία του κώδικα της εφαρμογής, χωρίς να έχουμε hard-coded τιμές. Σημειώνουμε ότι το Flask θέτει κάποιες μεταβλητές με τις αντίστοιχες τιμές τους από προεπιλογή σε

αυτό το λεξικό, όπως την μεταβλητή *APPLICATION_ROOT=΄/΄*, η οποία ενημερώνει την εφαρμογή σε ποια διαδρομή έχει τοποθετηθεί η εφαρμογή/διακομιστής ιστού.

Η πρώτη μεταβλητή, το λεγόμενο *«κρυφό κλειδί»*, είναι απαραίτητη ούτως ώστε οι συνεδρίες (*sessions*) από την πλευρά του πελάτη να διατηρούνται ασφαλείς. Η μεταβλητή αυτή ανήκει στις προκαθορισμένες μεταβλητές του Flask με τιμή *None*, συνεπώς πρέπει να δηλώσουμε κατηγορηματικά μία τιμή, την οποία οφείλουμε να έχουμε ρητά μυστική. Στο παραπάνω παράδειγμα ορίζουμε μια απλή τιμή *΄my-secret΄*. Η δεύτερη μεταβλητή ορίζει το μονοπάτι της βάσης δεδομένων για την εφαρμογή, το οποίο είναι ένα απλό αρχείο, όπως τύπου SQLite. Η συνάρτηση join του module os.path συνδέει ένα ή περισσότερα στοιχεία διαδρομής σε ένα πλήρες μονοπάτι ανεξαρτήτως του ΛΣ που χρησιμοποιούμε ενώ χειρίζεται κατάλληλα και τις τελικές καθέτους στο δοθέν μονοπάτι, εξαλείφοντας έτσι προβλήματα συμβατότητας και ελέγχων των χαρακτήρων οριοθέτησης. Το μέλος του αντικειμένου instance_path περιέχει το μονοπάτι όπου η εφαρμογή θα δημιουργεί π.χ. δυναμικά αρχεία.

Τα παραπάνω μοτίβα είναι αρκετά απλά και κατανοητά, αλλά καθώς η εφαρμογή μας μεγαλώνει, θα αντιμετωπίσουμε δυσκολίες κρατώντας όλο τον κώδικα σε ένα αρχείο. Ο κώδικας πρέπει να μοιράζεται σε περισσότερα αρχεία και υποφακέλους, ώστε να προωθηθεί η επαναχρησιμοποίηση και η αναγνωσιμότητα. Επίσης το μοτίβο αυτό θα μας οδηγήσει σε διάφορα προβλήματα όπως *κυκλικές εισαγωγές* (*circular imports*), επειδή η εφαρμογή μας εξαρτάται από το αντικείμενο app το οποίο και θα είναι καθολικής εμβέλειας, διότι χρειαζόμαστε να το εισάγουμε σε κάθε αρχείο που το χρειάζεται. Το Flask παρέχει ένα κοινό μοτίβο για την επίλυση αυτού του προβλήματος, που ονομάζεται *«εργοστάσιο εφαρμογής» (application factory)*. Τα εργοστάσια εφαρμογής σχετίζονται με το λεγόμενο *«πλαίσιο της εφαρμογής»* (*application context)*, το οποίο περιλαμβάνει τη συλλογή αρχείων Python και λειτουργικών μονάδων που συνθέτουν την εφαρμογή μας και τα ενώνει με τέτοιο τρόπο, έτσι ώστε να είναι ορατά το ένα στο άλλο. Για να δημιουργήσουμε ένα εργοστάσιο εφαρμογής, πρέπει να δημιουργήσουμε μια συνάρτηση ονόματι *create_app()*, όπου θα δημιουργείται η εφαρμογή και το αντικείμενο αυτής θα είναι η τιμή επιστροφής της. Επιπρόσθετα, χρησιμοποιώντας την παραπάνω συνάρτηση, έχουμε τη δυνατότητα να περάσουμε ως παράμετρο μια κλάση αποθηκευμένη σε ένα πρόσθετο αρχείο, η οποία θα περιέχει ως μέλη της όλες τις μεταβλητές διαμόρφωσης, απομονώνοντας ακόμα περισσότερο την λογική της εφαρμογής σε διάφορα κομμάτια.

Πριν παραθέσουμε τον κώδικα για το εργοστάσιο εφαρμογής, θα αναλύσουμε τις μεταβλητές διαμόρφωσης, ούτως ώστε να υπάρχει συμφωνία με την σειρά ανάγνωσης των αρχείων από την εφαρμογή. Στον κώδικα 4.7 παρουσιάζουμε την πλήρη κλάση που θα χρησιμοποιεί η εφαρμογή μας, κατά την εκκίνηση του διακομιστή. Για ορισμένες μεταβλητές, χρησιμοποιούμε το μοτίβο των μεταβλητών περιβάλλοντος, όπου το πρόγραμμά μας ψάχνει για ένα κρυφό αρχείο ονόματος *.env*, το οποίο και παρέχει τιμές για ορισμένες μεταβλητές για ανάγνωση. Εάν κάποια μεταβλητή δεν βρεθεί σε αυτό το αρχείο, θα παρέχουμε μια προκαθορισμένη τιμή, ώστε να είμαστε σίγουροι ότι ο διακομιστής θα χρησιμοποιήσει τις κατάλληλες τιμές παραμέτρων. Το μοτίβο που μόλις περιγράψαμε θα έχει την μορφή *<μεταβλητή> = os.environ.get(<όνομα_μεταβλητής>) or <προκαθορισμένη_τιμή>*.

Αναλυτικότερα, στον κώδικα 4.5 έχουμε την μεταβλητή BASE_PATH, η οποία αντιπροσωπεύει το απόλυτο μονοπάτι της εφαρμογής. Η μεταβλητή αυτή ορίζεται εκτός της κλάσης Config, καθώς θα χρησιμοποιείται μόνο από το αρχείο config.py. Η δημιουργία της γίνεται δυναμικά χρησιμοποιώντας τις συναρτήσεις dirname και abspath της ενότητας os.path, οι οποίες επιστρέφουν το όνομα του καταλόγου από την καθορισμένη διαδρομή και το απόλυτο μονοπάτι αντίστοιχα. Εμφωλεύοντας την μία στην άλλη και κατόπιν εμφωλεύοντας και την μαγική μεταβλητή *__file__*, λαμβάνουμε δυναμικά

την απόλυτη διαδρομή του καταλόγου όπου βρίσκεται το πρόγραμμα μας. Δεδομένου ότι η ενότητα config.py είναι το αρχείο όπου διαβάζονται όλες οι μεταβλητές περιβάλλοντος, πρόκειται να εισάγουμε το αρχείο .env πριν δημιουργηθεί η κλάση Config, έτσι ώστε οι μεταβλητές να έχουν ήδη οριστεί κατά τη δημιουργία της κλάσης. Αυτό επιτυγχάνεται με την χρήση της συνάρτησης load_dotenv της βιβλιοθήκης dotenv, στην οποία παρέχουμε το μονοπάτι της εφαρμογής και το αρχείο .env προς ανάγνωση. Σημειώνουμε ότι η εντολή flask εισάγει αυτόματα στο περιβάλλον τυχόν μεταβλητές που ορίζονται στο αρχείο .env. Το αρχείο αυτό, ωστόσο, πρόκειται να χρησιμοποιηθεί επίσης σε περιβάλλον production της εφαρμογής, η οποία δεν θα χρησιμοποιεί την εντολή flask. Για το λόγο αυτό, προτείνεται να εισάγουμε ρητά τα περιεχόμενα του αρχείου .env.

Ως μέλη της κλάσης Config, δηλώνουμε θεμελιώδεις μεταβλητές που θα χρησιμοποιηθούν από την εφαρμογή. Φυσικά προσθέτουμε και κάποια σχόλια τα οποία επιτρέπουν την ομαδοποίηση των μεταβλητών ανά το σκοπό που υπηρετούν, για καλύτερη αναγνωσιμότητα. Το πρώτο κομμάτι του κώδικα 4.5 περιέχει το ΕΑΠ της βάσης δεδομένων για την επικοινωνία αυτής με τον διακομιστή μας και τις παραμέτρους ρύθμισης για το ηλεκτρονικό ταχυδρομείο της εφαρμογής. Οι ενδιάμεσες είναι μεταβλητές σχετικές με τον διακομιστή μας, όπως το κρυφό κλειδί που εξηγήσαμε παραπάνω, η απενεργοποίηση ταξινόμησης κατά αλφαβητική σειρά των κλειδιών JSON, οι επιτρεπτοί τύποι αρχείων που μπορούν να αποθηκευτούν στην πλατφόρμα και τέλος δύο μεταβλητές σχετικές με την βιβλιοθήκη SQLAlchemy, ένας ΣΧΑ (ORM), η οποία συνοπτικά μας επιτρέπει να γράφουμε ερωτήματα SQL σε μορφή κώδικα python, με την χρήση κλάσεων και των μεθόδων τους.

Κώδικας 4.7: Διαμόρφωση παραμέτρων της εφαρμογής

```
import json
import os
from dotenv import load_dotenv

BASE_PATH = os.path.dirname(os.path.abspath(__file__))
load_dotenv(os.path.join(BASE_PATH, '.env'))

class Config(object):
      # ================= DB configuration ================= #
      user = os.environ.get('POSTGRES_USER') or 'postgres'
      password = os.environ.get('POSTGRES_PASSWORD') or 'postgres'
      host = os.environ.get('POSTGRES_HOST') or 'localhost'
      port = os.environ.get('POSTGRES_PORT') or 5432
      database = os.environ.get('POSTGRES_DB') or 'web_db'
      DATABASE_CONNECTION_URI =
f'postgresql+psycopg2://{user}:{password}@{host}:{port}/{database}'

      # ================= App configuration ================= #
      SECRET_KEY = os.environ.get('SECRET_KEY') or os.urandom(32)
      SQLALCHEMY_DATABASE_URI = DATABASE_CONNECTION_URI
      SQLALCHEMY_TRACK_MODIFICATIONS = False
      JSON_SORT_KEYS = False
```

```
    ALLOWED_EXTENSIONS = {'json'}

    # ================ Email config ================ #
    MAIL_SERVER = os.environ.get('MAIL_SERVER') or 'localhost'
    MAIL_PORT = os.environ.get('MAIL_PORT') or 25
    MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS') or False
    MAIL_USE_SSL = os.environ.get('MAIL_USE_SSL') or False
    MAIL_USERNAME = os.environ.get('MAIL_USERNAME') or False
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD') or False
   MAIL_DEFAULT_SENDER=os.environ.get('MAIL_DEFAULT_SENDER') or /
     'no-reply@wellfort.org'
    try:
    ADMINS = json.loads(os.environ.get('ADMINS'))
    except Exception:
    ADMINS = None

    SECURE_REPO_URL = os.environ.get('SECURE_REPO_URL') or
'https://localhost:5000'

    # ================ Elasticsearch ================ #
    ELASTICSEARCH_URL = os.environ.get('ELASTICSEARCH_URL')

    # ================ Redis ================ #
    REDIS_URL = os.environ.get('REDIS_URL') or 'redis://'
```

Το δεύτερο κομμάτι του κώδικα 4.7 περιέχει μεταβλητές τύπου URI. Η ομάδα παραμέτρων ηλεκτρονικού ταχυδρομείου θα χρησιμοποιηθεί για την αποστολή αυτοματοποιημένων email από την εφαρμογή. Οι υπόλοιπες είναι αναγκαίες για την επικοινωνία της εφαρμογής με τις διάφορες υπηρεσίες που χρησιμοποιεί η πλατφόρμα, όπως του ασφαλούς αποθετηρίου, όπως περιγράφηκε στην ενότητα 4.2, της Elasticsearch για αναζήτηση εγγραφών και του Redis για υποβολή ασύγχρονων εργασιών. Η χρησιμότητα των μεταβλητών αυτών θα εξηγηθεί λεπτομερώς αργότερα, καθώς θα παραθέτουμε και τον κώδικα της αντίστοιχης λειτουργικότητας.

Έχοντας λοιπόν υπόψιν την παραπάνω κλάση μεταβλητών διαμόρφωσης, μπορούμε τώρα να χρησιμοποιήσουμε ένα εργοστάσιο εφαρμογής, το οποίο θα την εισάγει ως παράμετρο και θα αποθηκεύει τις τιμές των μεταβλητών αυτών για χρήση αργότερα. Στον κώδικα 4.6 παρουσιάζεται το εργοστάσιο που χρησιμοποιεί η εφαρμογή μας. Αρχικά εισάγουμε τις ενσωματωμένες βιβλιοθήκες Python os για λειτουργίες σε επίπεδο ΛΣ και logging για λειτουργίες καταγραφής. Επίσης από το υποπακέτο *logging.handlers* εισάγουμε τους δομητές *SMTPHandler* για την ρύθμιση ηλεκτρονικού ταχυδρομείου και *RotatingFileHandler* για κυκλική εγγραφή καταγραφής διαφόρων συμβάντων. Κατόπιν εισάγουμε τις βιβλιοθήκες τρίτων που χρησιμοποιούμε στην εφαρμογή, όπως το Flask, Elasticsearch και διάφορες επεκτάσεις τύπου Flask, οι οποίες έχουν την μορφή *flask-<όνομα>*. Οι επεκτάσεις Flask (*Flask extensions*) είναι πακέτα που προσθέτουν επιπλέον λειτουργικότητα στην εφαρμογή. Παρακάτω χρησιμοποιούμε τις ακόλουθες επεκτάσεις Flask:

- SQLAlchemy. Είναι ένα φιλικό Flask-wrapper για το δημοφιλές πακέτο SQLAlchemy, το οποίο όπως αναφέραμε είναι ένας ΣΧΑ. Οι ΣΧΑ επιτρέπουν στις εφαρμογές να διαχειρίζονται μια βάση δεδομένων χρησιμοποιώντας οντότητες υψηλού επιπέδου όπως κλάσεις, αντικείμενα και μεθόδους αντί για πίνακες και SQL. Η δουλειά τους είναι να μεταφράζει τις λειτουργίες υψηλού επιπέδου σε εντολές βάσης δεδομένων.
- Migrate. Χειρίζεται τις μετεγκαταστάσεις της βάσης δεδομένων για εφαρμογές Flask. Η προσθήκη ή τροποποίηση πινάκων ή στηλών αποτελεί ένα καινούριο στιγμιότυπο του σχήματος της βάσης και με την επέκταση αυτή μπορούμε να εφαρμόσουμε το επιθυμητό σχήμα, το οποίο καλύπτει τις ανάγκες της εφαρμογής στην δεδομένη στιγμή.
- Bootstrap. Παρέχει ένα βασικό template έτοιμο προς χρήση, το οποίο αναλαμβάνει την εισαγωγή του Bootstrap.
- Login. Παρέχει λειτουργικότητα σχετικά με την συνεδρία χρήστη. Διαχειρίζεται τις συνήθεις εργασίες σύνδεσης (*login*), αποσύνδεσης (*logout*) και απομνημόνευσης των συνεδριών.
- Mail. Παρέχει μια απλή διεπαφή για ρύθμιση του SMTP με την εφαρμογή μας και για την αποστολή μηνυμάτων από τις διάφορες διαδρομές της.
- DebugToolbar. Προσθέτει μια επικάλυψη γραμμής εργαλείων σε όλες τις διεπαφές της εφαρμογής, όταν αυτή βρίσκεται σε λειτουργία αποσφαλμάτωσης (*Debug mode*). Στην γραμμή εργαλείων περιέχονται χρήσιμες πληροφορίες για τον εντοπισμό σφαλμάτων, όπως αιτήματα στην βάση δεδομένων και εμφάνιση των κεφαλίδων αιτημάτων και απόκρισης HTTP.

Τέλος η βιβλιοθήκη *sqlalchemy_utils* παρέχει προσαρμοσμένους τύπους δεδομένων και διαφορετικές βοηθητικές συναρτήσεις για το πακέτο SQLAlchemy. Η επόμενη ομάδα εισαγωγών είναι σχετική με αρχεία που ανήκουν στην εφαρμογή μας. Έχοντας ως οδηγό την δομή που παρουσιάζεται στο σχήμα 4.1, εισάγουμε την κλάση μεταβλητών διαμόρφωσης Config, την οποία χρησιμοποιούμε ως προκαθορισμένη τιμή της παραμέτρου *config_class* στο εργοστάσιό μας και την συνάρτηση *register_routes()* από το αρχείο routes.py, η οποία ενθυλακώνει την λειτουργικότητα για την εγγραφή των διαδρομών της ΔΧ και της ΔΠΕ στην εφαρμογή Flask.

Στον κύριο μέρος του αρχείου αρχικοποιούμε πρώτα όλους τους δομητές από τις επεκτάσεις Flask. Καθώς χρησιμοποιούμε το μοτίβο εργοστασίου εφαρμογής, πρέπει να καλέσουμε την μέθοδο *init_app()* σε κάθε αντικείμενο που αρχικοποιήσαμε εντός της συνάρτησης, έτσι ώστε να συσχετίσουμε την εφαρμογή Flask με την αντίστοιχη επέκταση. Με αυτόν τον τρόπο το αντικείμενο κάθε επέκτασης μπορεί να εισαχθεί οποιαδήποτε στιγμή και όχι μόνο κατόπιν της δημιουργίας της εφαρμογής.

Κώδικας 4.8: "Εργοστάσιο" Εφαρμογής

```
import logging
import os

from logging.handlers import RotatingFileHandler, SMTPHandler

from elasticsearch import Elasticsearch
from flask import Flask
from flask_bootstrap import Bootstrap
from flask_debugtoolbar import DebugToolbarExtension
from flask_mail import Mail
```

```
from flask_migrate import Migrate
from flask_profiler import Profiler
from redis import Redis
from rq import Queue
from sqlalchemy_utils.functions import create_database, database_exists

from app.routes import register_routes
from config import Config

bootstrap = Bootstrap()
db = SQLAlchemy()
debug_toolbar = DebugToolbarExtension()
mail = Mail()
migrate = Migrate()

def create_app(config_class=Config):
    # ======= App instantiation ======= #
    app = Flask(__name__)
    app.config.from_object(config_class)

    # ======= DB instantiation ======= #
    if not database_exists(str(app.config['DATABASE_CONNECTION_URI'])):
        create_database(str(app.config['DATABASE_CONNECTION_URI']))
    from app.db import db
    db.init_app(app)
    migrate.init_app(app, db)

    # ======= Bootstrap instantiation ======= #
    bootstrap.init_app(app)

    # ======= Login instantiation ======= #
    from app.login import login
    login.init_app(app)

    # ======= Mail instantiation ======= #
    mail.init_app(app)

    # ======= Debug Toolbar instantiation ======= #
    debug_toolbar.init_app(app)

    app.elasticsearch = Elasticsearch(app.config['ELASTICSEARCH_URL']) \
        if app.config['ELASTICSEARCH_URL'] else None

    app.redis = Redis.from_url(app.config['REDIS_URL'])
    app.task_queue = Queue('exp-web-tasks', connection=app.redis)
```

```
    # ======= Routes, errors registration ======= #
    register_routes(app)

    if app.debug or app.testing:
        Profiler(app)
    else:
        #  =============== Errors Mail Handler =============== #
        if app.config['MAIL_SERVER']:
            auth = None
            if app.config['MAIL_USERNAME'] or
app.config['MAIL_PASSWORD']:
                auth = (app.config['MAIL_USERNAME'],
app.config['MAIL_PASSWORD'])
            secure = None
            if app.config['MAIL_USE_TLS']:
                secure = ()

            mail_handler = SMTPHandler(
                mailhost=(app.config['MAIL_SERVER'],
app.config['MAIL_PORT']),
                fromaddr=app.config['MAIL_DEFAULT_SENDER'],
                toaddrs=app.config['ADMINS'],
                subject='Web Application - Runtime Error',
                credentials=auth,
                secure=secure
            )
            mail_handler.setLevel(logging.ERROR)
            app.logger.addHandler(mail_handler)
        # ======= Logging config ======= #
        os.makedirs('logs', exist_ok=True)

        app_handler = RotatingFileHandler('logs/app.log',
maxBytes=100000)
        app_handler.setFormatter(
            logging.Formatter('%(asctime)s %(levelname)s: %(message)s [in
%(pathname)s:%(lineno)d]')
        )
        app_handler.setLevel(logging.INFO)
        app.logger.addHandler(app_handler)

        app.logger.setLevel(logging.INFO)
        app.logger.info('Web-service started')

    return app
```

Στο σώμα της συνάρτησης εργοστασίου δηλώνουμε την εφαρμογή μας και τις παραμέτρους ρύθμισης που θα χρησιμοποιήσει σύμφωνα με την κλάση που έχει δοθεί ως παράμετρος στην κλήση της, εάν υπάρχει. Κατόπιν χρησιμοποιούμε 2 συναρτήσεις από το πακέτο sqlalchemy_utils. Πρώτα γίνεται έλεγχος για την ύπαρξη της βάσης δεδομένων σύμφωνα με το URI που δόθηκε ως παράμετρος ρύθμισης, με την βοήθεια της συνάρτησης *database_exists()*. Εάν η συνθήκη είναι ψευδής, τότε καλούμε την δεύτερη συνάρτηση *create_database()*, η οποία αναλαμβάνει και την δημιουργία της βάσης. Με αυτόν τον τρόπο η εφαρμογή αναλαμβάνει την δημιουργία της βάσης για εμάς. Έπειτα συνδέουμε τις επεκτάσεις με την εφαρμογή, όπως εξηγήσαμε παραπάνω. Στη συνέχεια συνδέουμε στο αντικείμενο app τα μέλη elasticseach, redis και task_queue, τα οποία αρχικοποιούνται καλώντας τον αντίστοιχο δομητή της βιβλιοθήκης με παράμετρο το URL που αναγιγνώσκονται από την κλάση Config κατά την εκκίνηση της εφαρμογής. Συνδέοντας ένα μέλος στο αντικείμενο app μπορεί να φαίνεται λίγο περίεργο, αλλά τα αντικείμενα στην Python δεν είναι αυστηρά στην δομή τους και συνεπώς μπορούμε να προσθέσουμε νέα μέλη/χαρακτηριστικά ανά πάσα στιγμή. Η προσαρμοσμένη συνάρτηση register_routes() του αρχείου routes.py αναλαμβάνει την καταχώριση όλων των διαδρομών για την εφαρμογή, η οποία θα παρατεθεί στην επόμενη υποενότητα.

Το εργοστάσιο της εφαρμογής μας ολοκληρώνεται με την συνθήκη if-else όπου ελέγχεται εάν η εφαρμογή μας είναι σε λειτουργία development ή testing, ή λειτουργία production. Εάν η πρώτη συνθήκη είναι αληθής, τότε αρχικοποιούμε μια ακόμη επέκταση, το flask-profiler, η οποία θα μας βοηθήσει στην μέτρηση του χρόνου αποκρίσεων κάθε διαδρομής, έτσι ώστε να αξιολογήσουμε την απόδοση του κώδικα. Εάν η εφαρμογή είναι σε λειτουργία production, τότε προχωράμε στην δημιουργία ενός καταγραφέα (*logger*) SMTP, ο οποίος θα στέλνει αυτοματοποιημένα μηνύματα ηλεκτρονικού ταχυδρομείου στις διευθύνσεις των προγραμματιστών, όταν λαμβάνει χώρα στην εφαρμογή κάποιο σφάλμα. Η ρύθμιση αυτού είναι κάπως εκτεταμένη, λόγω της ανάγκης χειρισμού προαιρετικών επιλογών ασφαλείας που υπάρχουν σε πολλούς διακομιστές email. Στην ουσία, ο κώδικας δημιουργεί το αντικείμενο SMTPHandler, ορίζει το επίπεδο του έτσι ώστε να αναφέρει μόνο σφάλματα και όχι προειδοποιήσεις, πληροφοριακά μηνύματα ή μηνύματα αποσφαλμάτωσης και τέλος, τον επισυνάπτει στο αντικείμενο app.logger του Flask, για να είναι ορατός στην εφαρμογή. Η λήψη αναφοράς σφαλμάτων μέσω email είναι χρήσιμη αλλά πολλές φορές όχι αρκετή. Υπάρχουν ορισμένες συνθήκες αποτυχίας που δεν καταλήγουν σε εξαίρεση Python, αλλά μπορεί να εξακολουθούν να είναι ενδιαφέρουσες για αποθήκευση για σκοπούς αποσφαλμάτωσης. Γι’ αυτό το λόγο, χρειαζόμαστε δημιουργία αρχείων καταγραφής για τα οποία το δεύτερο σκέλος της συνθήκης else είναι υπεύθυνο. Για να ενεργοποιήσουμε ένα αρχείο καταγραφής, αυτή τη φορά τύπου RotatingFileHandler, πρέπει να προσαρτήσουμε τον χειριστή (handler) στον καταγραφέα app.logger της εφαρμογής, όπως κάναμε και με τον χειριστή email. Τα αρχεία καταγραφής θα πρέπει να ανήκουν σε ξεχωριστό φάκελο, ελέγχοντας την ύπαρξη του οποίου με την εντολή os.makedirs() περνώντας την παράμετρο *exist_ok=True*. Εάν ο φάκελος δεν υπάρχει κατά την εκκίνηση της εφαρμογής θα δημιουργηθεί, εάν ναι, τότε η εντολή δεν θα έχει κάποιο αποτέλεσμα. Η παράμετρος *exist_ok=True* είναι σημαντική, καθώς αν την παραλείψουμε και ο φάκελος υπάρχει, τότε θα οδηγηθούμε σε εξαίρεση. Η κλάση RotatingFileHandler είναι χρήσιμη καθώς περιστρέφει τα αρχεία καταγραφής, διασφαλίζοντας ότι αυτά δεν μεγαλώνουν πολύ όταν η εφαρμογή εκτελείται για μεγάλο χρονικό διάστημα. Σε αυτήν την περίπτωση, περιορίζουμε το μέγεθος στα 10KB και διατηρούμε τα τελευταία δέκα αρχεία καταγραφής ως αντίγραφο ασφαλείας. Η κλάση logging.Formatter παρέχει προσαρμοσμένη μορφοποίηση για τα μηνύματα καταγραφής. Επειδή αυτά τα μηνύματα πηγαίνουν σε ένα αρχείο, θέλουμε να έχουν όσο το δυνατόν περισσότερες πληροφορίες. Επομένως, χρησιμοποιούμε μια μορφή που περιλαμβάνει τη χρονική σήμανση, το επίπεδο καταγραφής, το μήνυμα και το αρχείο προέλευσης και τον αριθμό γραμμής από όπου προήλθε η καταχώριση η

καταχώρηση καταγραφής. Για να κάνουμε την καταγραφή πιο χρήσιμη, χαμηλώνουμε επίσης το επίπεδο καταγραφής στην κατηγορία INFO, τόσο στον καταγραφέα της εφαρμογής όσο και στον καταγραφέα αρχείων. Όταν αυτή η εφαρμογή εκτελείται σε λειτουργία production, αυτές οι εγγραφές καταγραφής θα μας ενημερώσουν πότε έγινε επανεκκίνηση του διακομιστή.

4.2.2.4 Καταχώριση διαδρομών

Όπως αναφέραμε στην προηγούμενη υποενότητα, δημιουργήσαμε μια προσαρμοσμένη συνάρτηση ονόματι register_routes(), η οποία αναλαμβάνει την καταχώρηση όλων των διαδρομών της εφαρμογής που θα υλοποιήσουμε. Η υλοποίηση της συνάρτησης αυτής απεικονίζεται στον Κώδικα 4.11. Πριν προχωρήσουμε στην επεξήγησή του, κρίνεται αναγκαίο να αναλύσουμε την χρήση σχεδιαγραμμάτων (Blueprints). Στην παρακάτω υποενότητα παραθέτουμε την χρησιμότητά τους.

4.2.2.4.1 Σχεδιαγράμματα (Blueprints)

Το Flask μας παρέχει την δυνατότητα χρήσης "σχεδιαγραμμάτων" (*Blueprints*), τα οποία είναι λογικές δομές που αντιπροσωπεύουν ένα υποσύνολο της εφαρμογής. Ένα Blueprint μπορεί να περιλαμβάνει στοιχεία, όπως διαδρομές, συναρτήσεις προβολών (view functions), φόρμες, templates και στατικά αρχεία. Γράφοντας ένα blueprint σε ένα ξεχωριστό πακέτο Python, επιτυγχάνουμε την ενσωμάτωση και ομαδοποίηση στοιχείων που σχετίζονται με συγκεκριμένες λειτουργίες-χαρακτηριστικά της εφαρμογής. Τα περιεχόμενα ενός blueprint είναι αρχικά σε "αδρανή" (dormant) κατάσταση. Για να συσχετιστούν αυτά τα περιεχόμενα, το blueprint πρέπει να καταχωρηθεί στην εφαρμογή. Κατά την εγγραφή, όλα τα στοιχεία που προστέθηκαν στο blueprint περνούν στην εφαρμογή. Μπορούμε λοιπόν να θεωρήσουμε το blueprint ως μια προσωρινή αποθήκευση για συγκεκριμένες λειτουργίες της εφαρμογής που βοηθάει στην οργάνωση του κώδικά μας.

Η δημιουργία ενός blueprint είναι αρκετά παρόμοια με τη δημιουργία όλης της εφαρμογής. Αυτό γίνεται στο module __init__.py του πακέτου όπου δηλώνεται το blueprint. Ο κώδικας 4.9 παρουσιάζει την αρχικοποίηση ενός blueprint, υπεύθυνο για την εγκατάσταση και παρουσίαση προσαρμοσμένων σελίδων για σφάλματα (π.χ 404 not found), έτσι ώστε οι χρήστες να μην βλέπουν τις απλές προεπιλεγμένες σελίδες. Ο κώδικας 4.10 παρουσιάζει τους χειριστές σφαλμάτων για λόγους συνέχειας με τον κώδικα 4.9.

Κώδικας 4.9: *web/app/errors/__init__.py* - Οργάνωση σφαλμάτων UI σε Blueprint

```
from flask import Blueprint

errors_bp = Blueprint(
    'errors',
    __name__
)

from app.errors.handlers import register_errors
register_errors(errors_bp)
```

Η κλάση Blueprint λαμβάνει το όνομα του blueprint, το όνομα του βασικού module (συνήθως ορίζεται σε __name__ όπως στην περίπτωση της κλήσης του δομητή Flask) και άλλες διάφορες παραμέτρους,

που δεν χρειάζονται σε αυτήν την απλή περίπτωση. Αφού δημιουργηθεί το αντικείμενο blueprint, εισάγουμε τη συνάρτηση register_errors(), η οποία αναλαμβάνει την καταχώρηση σελίδων σφαλμάτων για το χρήστη. Η εισαγωγή πραγματοποιείται κατόπιν της δήλωσης του blueprint για την αποφυγή κυκλικών εισαγωγών.

Κώδικας 4.10: *web/app/errors/handlers.py* - Υλοποίηση σελίδων σφαλμάτων

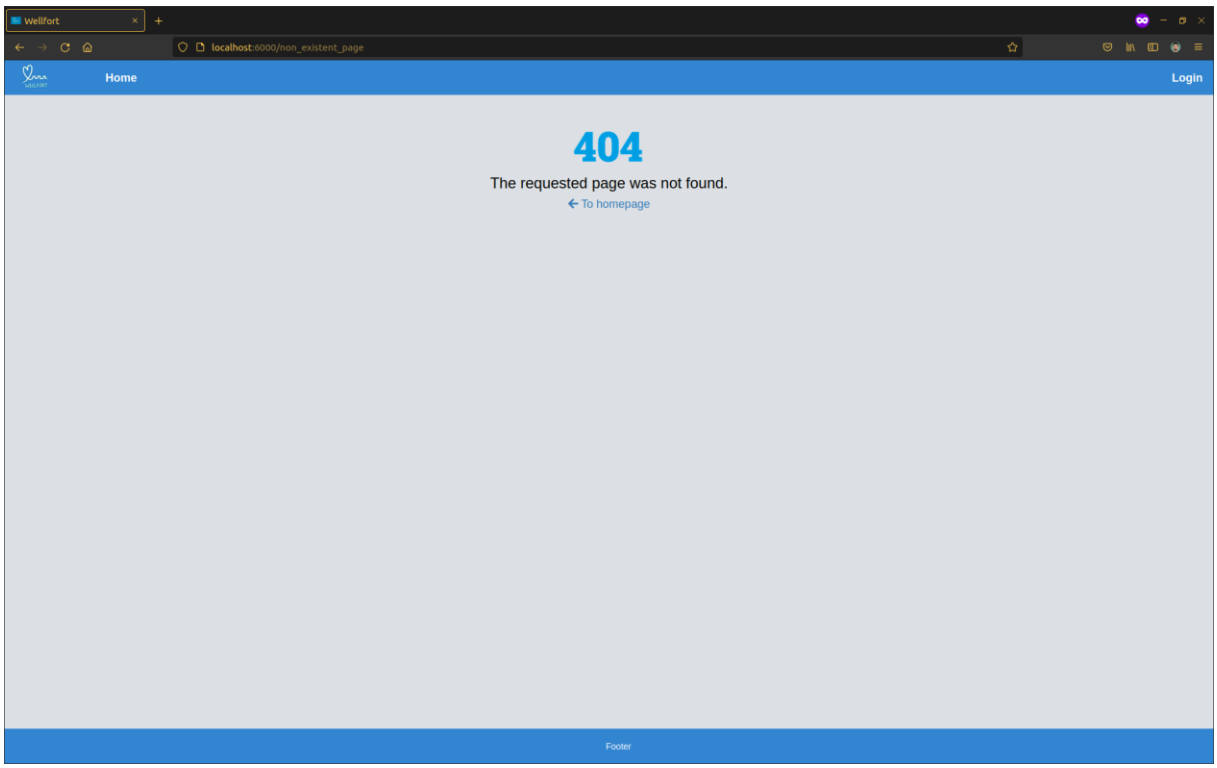
```
from flask import Blueprint, render_template, request


# ================== Errors Registration ================== #
def register_errors(bp: Blueprint):
    @bp.app_errorhandler(400)
    def bad_request_error(error):
        return render_template('errors/400.html',
message=error.description), 400

    @bp.app_errorhandler(401)
    def unauthorized_error(error):
        return render_template('errors/401.html'), 401

    @bp.app_errorhandler(404)
    def not_found_error(error):
        # 404 error is not caught by the custom api error handler class
        # as the API handler occurs only on the exact routes
        if request.path.startswith('/api/'):
            return dict(status=error.code, message=error.description),
error.code
        return render_template('errors/404.html'), 404

    @bp.app_errorhandler(405)
    def method_not_allowed_error(error):
        return render_template('errors/405.html'), 405

    @bp.app_errorhandler(500)
    def internal_error(error):
        return render_template('errors/500.html'), 500

    @bp.app_errorhandler(503)
    def service_unavailable_error(error):
        return render_template('errors/503.html'), 503
```

Στον κώδικα 4.9 παρουσιάζουμε την συνάρτηση register_errors(), η οποία περιέχει τους προσαρμοσμένους χειριστές λαθών, ως εμφωλευμένες συναρτήσεις. Η αντίστοιχη υπο-συνάρτηση ενεργοποιείται σύμφωνα με τον κωδικό HTTP που έχει επιστρέψει κάποιο μέρος του κώδικα. Αυτό επιτυγχάνεται με τον διακοσμητή "@<*blueprint_instance*>.app_errorhandler(<*HTTP_status*>)", ο

οποίος προηγείται κάθε δήλωσης της αντίστοιχης υπο-συνάρτησης. Ο κώδικας αναλαμβάνει τα πιο συνήθη σφάλματα που μπορούν να λάβουν χώρα στην πλατφόρμα, τα οποία είναι:

- Bad Request (400): Ο χρήστης έχει πραγματοποιήσει κάποιο εσφαλμένο ερώτημα ή έχει παραλείψει υποχρεωτικά πεδία σε υποβολή φόρμας.
- Unauthorized (401): Ο χρήστης προσπαθεί να αποκτήσει πρόσβαση σε προστατευμένο πόρο, χωρίς να έχει ταυτοποιηθεί.
- Not found (404): Ο ζητούμενος πόρος δεν υπάρχει στην πλατφόρμα.
- Method not allowed (405): Υποδεικνύει ότι η μέθοδος HTTP αιτήματος ελήφθη από τον διακομιστή, αλλά ο διακομιστής δεν επιτρέπει τη συγκεκριμένη μέθοδο για το ζητούμενο πόρο.
- Internal Server Error (500): Γενική απόκριση σφάλματος. Υποδεικνύει ότι ο διακομιστής αντιμετώπισε κάποια συνθήκη, η οποία τον εμπόδισε από το να εκπληρώσει το αίτημα. Συχνά το σφάλμα αυτό πυροδοτείται κατά την "ρίψη" απροσδόκητης εξαίρεσης Python.
- Service Unavailable (503): Υποδηλώνει ότι ο διακομιστής δεν μπορεί να χειριστεί προσωρινά το αίτημα. Στην περίπτωση μας πυροδοτείται, όταν κάποια υπηρεσία της πλατφόρμας δεν είναι διαθέσιμη, π.χ. το Ασφαλές Αποθετήριο Δεδομένων.

Το Flask μας παρέχει την υλοποιημένη συνάρτηση render_template(), η οποία επιστρέφει απόκριση για σελίδες HTML. Ως βασική παράμετρο περνάμε τη διαδρομή στην προσαρμοσμένη σελίδα HTML που έχουμε δημιουργήσει και τυχόν άλλες μεταβλητές ως ορίσματα λέξεων-κλειδιών (*keyword arguments*) τις οποίες θέλουμε να είναι προσβάσιμες κατά το rendering της σελίδας. Στην εικόνα 4.1 παρουσιάζουμε στιγμιότυπο της εφαρμογής, όταν ζητηθεί μη υπάρχουσα διαδρομή.



Εικόνα 4.1: Απεικόνιση μη υπάρχουσας σελίδας

4.2.2.4.2 Εγγραφή σχεδιαγραμμάτων στην εφαρμογή

Η κάθε μία λειτουργικότητα της εφαρμογής θα υπόκειται και σε διαφορετικό blueprint, για τους λόγους που αναλύσαμε παραπάνω. Η εφαρμογή πρέπει να γνωρίζει για την ύπαρξη των σχεδιαγραμμάτων, για αυτό το λόγο πρέπει να τα εγγράψουμε κατά την αρχικοποίηση της.

Κώδικας 4.11: Καταχώριση διαδρομών της εφαρμογής

```
import os

from flask import Flask, send_from_directory, render_template, request,
url_for
from flask_login import login_required

from config import EXPORT_FOLDER
from app.db import db
from app.models import Query, Task


def register_routes(app: Flask):
    # ======= UI routes registration ======= #
    from app.query import query_bp
    app.register_blueprint(query_bp)
    from app.auth import auth_bp
    app.register_blueprint(auth_bp)
    from app.errors import errors_bp
    app.register_blueprint(errors_bp)

    # ======= API v1 registration ======= #
    from app.api.v1 import api_v1_bp
    app.register_blueprint(api_v1_bp)

    @app.route('/')
    @app.route('/index')
    @login_required
    def index():
        return render_template('index.html', queries=Query.query)

   @app.route('/favicon.ico')
    def favicon():
        return send_from_directory(
            os.path.join(app.root_path, 'static'),
            'logo.ico',
            mimetype='image/x-icon'
        )
```

```
    @app.route('/files/<string:filename>')
    def download(filename):
        _dir =
Task.query.filter_by(id=request.args.get('ref')).first_or_404('File not
found').directory
        return send_from_directory(EXPORT_FOLDER, os.path.join(_dir,
filename), as_attachment=True)
```

Η εγγραφή επιτυγχάνεται με την προκαθορισμένη μέθοδο register_blueprint() η οποία ανήκει στο αντικείμενο της εφαρμογής που αρχικοποιήσαμε. Ως παράμετρο περνάμε το αντικείμενο blueprint που έχουμε αρχικοποιήσει στο αντίστοιχο πακέτο. Φυσικά πρέπει πρώτα να το εισάγουμε από το αντίστοιχο αρχείο. Στην συνάρτηση register_routes(), πραγματοποιούμε τις εγγραφές όλων των blueprint της εφαρμογής, κατόπιν της εισαγωγής τους. Τα blueprint που χρησιμοποιεί η εφαρμογή μας είναι τα ακόλουθα:

- *query_bp*: σχετίζεται με την υποβολή ερωτημάτων από τον αναλυτή. Σε αυτό ανήκουν οι διαδρομές αναζήτησης δεδομένων, παρουσίασης αποτελεσμάτων και υποβολής πειράματος.
- *auth_bp*: υπεύθυνο για τις διαδρομές αυθεντικοποίησης του χρήστη, όπως εγγραφή, σύνδεση και αποσύνδεση στην πλατφόρμα.
- *errors_bp*: αναλαμβάνει τις προσαρμοσμένες σελίδες σφαλμάτων, τις οποίες εμβαθύναμε στην υποενότητα 4.2.2.4.1
- *api_v1_bp*: περιλαμβάνει όλες τις διαδρομές της ΔΠΕ έκδοσης 1 της πλατφόρμας.

Παρακάτω θα αναλυθεί ξεχωριστά η κάθε μια από τις παραπάνω εναπομείναντες λειτουργικότητες. Κατόπιν δηλώνουμε τον root ΕΕΠ για τη διεπαφή χρήστη, η οποία θα είναι μόνη της καθώς δεν σχετίζεται με κάποια από τις παραπάνω λειτουργικότητες. Για να δηλώσουμε την διαδρομή χρησιμοποιούμε τον διακοσμητή @<app_instance>.route(<route>). Η διαδρομή root θα είναι ορατή και σε περίπτωση ‘/’ αλλά και σε περίπτωση ‘/index’, γι’ αυτό και δηλώνουμε τον διακοσμητή 2 φορές με τις αντίστοιχες τιμές. Ο διακοσμητής @login_required διασφαλίζει ότι ο πόρος θα είναι μόνο διαθέσιμος όταν ο χρήστης έχει συνδεθεί στην πλατφόρμα. Στη συνέχεια δηλώνουμε την διαδρομή /favicon.ico με την αντίστοιχη συνάρτηση, η οποία εμφανίζει το γνωστό favicon. Τα σύγχρονα προγράμματα περιήγησης εμφανίζουν αυτό το εικονίδιο στα αριστερά του τίτλου του εγγράφου και λαμβάνεται συνήθως από τον ΕΕΠ *website.com/favicon.ico* . Με την προκαθορισμένη συνάρτηση send_from_directory() εξυπηρετούμε με ασφάλεια διαδρομές που ζητήθηκαν μέσα από έναν κατάλογο. Σε αυτήν την περίπτωση περνάμε ως παράμετρο τη διαδρομή root της εφαρμογής σε συνδυασμό με τον κατάλογο των στατικών αρχείων ακολουθούμενη από την παράμετρο του ονόματος του favicon αρχείου. Τέλος περνάμε την παράμετρο mimetype με την τιμή που χρησιμοποιείται γενικώς για τα favicon. Επισημαίνουμε πως ένας τύπος MIME είναι μια ετικέτα που χρησιμοποιείται για τον προσδιορισμό ενός τύπου δεδομένων. Εξυπηρετεί τον ίδιο σκοπό στο διαδίκτυο με τις επεκτάσεις αρχείων που υπάρχουν π.χ. στα Microsoft Windows. Η διαδρομή */files/<filename>* αποσκοπεί στη λήψη αρχείων από δυναμικούς καταλόγους που δημιουργούνται κατά τη σύνθεση ή ανωνυμοποίηση των δεδομένων. Περισσότερες λεπτομέρειες στην υποενότητα 4.2.2.6.4.

4.2.2.5 Υλοποίηση σχήματος βάσης δεδομένων

Τα δεδομένα που θα αποθηκεύονται στην βάση δεδομένων θα αντιπροσωπεύονται από μια συλλογή κλάσεων, που ονομάζονται *Μοντέλα Βάσεων Δεδομένων* (*Database Models*). Το “στρώμα” ΣΧΑ του

πακέτου SQLAlchemy θα κάνει τις μεταφράσεις που απαιτούνται για να αντιστοιχίσει αντικείμενα τα οποία δημιουργούνται από αυτές τις κλάσεις σε σειρές στους κατάλληλους πίνακες της ΒΔ. Αρχικά η βάση δεδομένων πρέπει να αντιπροσωπεύεται από ένα αντικείμενο SQLAlchemy. Το αντικείμενο αυτό θα περιλαμβάνεται σε ξεχωριστό module, καθώς θα καλείται από όλα τα μοντέλα, τα οποία θα υλοποιηθούν ένα κατά module. Ο κώδικας 4.12 παρουσιάζει μια απλή τέτοια υλοποίηση.

Αφού αρχικοποιήσουμε το αντικείμενο καλώντας το δομητή της SQLAlchemy(), πρέπει να δώσουμε το σήμα επισύναψης του αντικειμένου αυτού στην εφαρμογή μας με το μοτίβο <*instance*>.init_app(<*app_instance*>), το οποίο και πραγματοποιήσαμε στον κώδικα 4.6, στη δημιουργία του εργοστασίου της εφαρμογής. Κατόπιν παραθέτουμε τέσσερις βοηθητικές συναρτήσεις, οι οποίες αντικατοπτρίζουν τις βασικές λειτουργίες δημιουργίας, ενημέρωσης και διαγραφής.

Κώδικας 4.12: *web/app/db.py* - Αντικείμενο της ΒΔ με βοηθητικές συναρτήσεις

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

def add(model, **kwargs):
    instance = model(**kwargs)
    db.session.add(instance)
    commit()

def edit(model, _id, **kwargs):
    instance = model.query.filter_by(id=_id).first()
    for attr, new_value in kwargs.items():
        setattr(instance, attr, new_value)
    commit()

def delete(model, _id):
    model.query.filter_by(id=_id).delete()
    commit()

def commit():
    db.session.commit()
```

Η συνάρτηση add() παίρνει ως παράμετρο το όνομα της κλάσης και τυχόν περαιτέρω ορίσματα λέξεων-κλειδιών τα οποία θα προωθηθούν στον δομητή αυτής. Οι αλλαγές στη ΒΔ γίνονται στο πλαίσιο μιας συνεδρίας ΒΔ, την οποία προσπελαύνουμε με την εντολή db.session. Γενικότερα πολλαπλές αλλαγές μπορούν να συγκεντρωθούν σε μια συνεδρία και μόλις καταχωρηθούν, μπορούμε να χρησιμοποιήσουμε την εντολή db.session.commit() για να αντικατοπτριστούν οι εκκρεμείς αυτές αλλαγές στην ΒΔ. Η κλήση στη συνάρτηση commit() πραγματοποιεί ακριβώς το παραπάνω, την οποία ορίζουμε με σύντομο όνομα για ευκολία και αναγνωσιμότητα. Η συνάρτηση edit() είναι υπεύθυνη για την ενημέρωση των εγγραφών. Λαμβάνει ως παράμετρο επίσης το όνομα της κλάσης αλλά και το μοναδικό αναγνωριστικό της εγγραφής προς ενημέρωση. Ακόμη η συνάρτηση αναμένει πρόσθετες παραμέτρους ως λέξεις-κλειδιά, οι οποίες αντιπροσωπεύουν τις στήλες της εγγραφής με την αντίστοιχη τιμή προς ενημέρωση.

Τονίζουμε πως κάθε μοντέλο κληρονομεί την ιδιότητα query από την κλάση Model της SQLAlchemy, την οποία "αλυσιδώνοντας" με την μέθοδο filter_by() μας επιστρέφει αντικείμενο τύπου Query, το οποίο ουσιαστικά αντιπροσωπεύει τα αποτελέσματα του SQL ερωτήματος. Η μέθοδος filter_by() αντιπροσωπεύει το φίλτρο ισότητας (WHERE …=...). Για να ολοκληρωθεί το ερώτημα, πρέπει να ορίσουμε το πλήθος των αποτελεσμάτων που επιθυμούμε. Αλυσιδώνοντας όλα τα παραπάνω με την μέθοδο first() ανακτούμε το πρώτο αντικείμενο (γραμμή του πίνακα) που εκπληρώνει το δοθέν φίλτρο (η παράμετρος που δίνεται στην μέθοδο filter_by()). Εφόσον βρεθεί η γραμμή σύμφωνα με το δοθέν αναγνωριστικό που δόθηκε ως παράμετρος, χρειαζόμαστε μια επανάληψη καθώς οι παράμετροι λέξεων-κλειδιών είναι αορίστου πλήθους. Η μέθοδος items() είναι ενσωματωμένη από την Python3 για τα λεξικά, όπου με την πρώτη μεταβλητή προσπελαύνουμε τα κλειδιά του λεξικού και με την δεύτερη την τιμή που αντιστοιχεί σε αυτό το κλειδί. Η ενσωματωμένη συνάρτηση setattr() ορίζει την τιμή (τρίτη παράμετρος) της καθορισμένης ιδιότητας (δεύτερη παράμετρος) του καθορισμένου αντικειμένου (πρώτη παράμετρος). Εδώ λοιπόν περνάμε ως παράμετρο το αντικείμενο τύπου χρήστη που βρέθηκε και για κάθε κλειδί θέτουμε την τιμή της εκάστοτε ιδιότητας. Φυσικά καλούμε την συνάρτηση commit() ώστε να αποθηκευτούν οι αλλαγές. Η συνάρτηση delete() λαμβάνει επίσης το όνομα της κλάσης ως παράμετρο και το μοναδικό αναγνωριστικό της εγγραφής προς διαγραφή. Το μοτίβο που ακολουθούμε είναι ίδιο με αυτό της συνάρτησης edit(), με την διαφορά ότι στο τελευταίο κομμάτι της αλυσίδας χρησιμοποιούμε την μέθοδο delete(), η οποία διαγράφει το αντικείμενο (ουσιαστικά την γραμμή). Ολοκληρώνουμε πάλι καλώντας την συνάρτηση commit().

Σύμφωνα με την υποενότητα 3.3.5, όπου και απεικονίσαμε τη δομή των πινάκων, θα παρουσιάσουμε μέσω μικρών κομματιών κώδικα την υλοποίηση τους. Για λόγους οργάνωσης και αναγνωσιμότητας , σημειώνουμε ότι όλα τα μοντέλα περιέχονται στο πακέτο web/models. Ο κώδικας 4.13 παρουσιάζει το μοντέλο για τον πίνακα των χρηστών της βάσης δεδομένων. Για να έχουμε πρόσβαση στα χαρακτηριστικά ενός μοντέλου η κλάση μας πρέπει να κληρονομεί τις ιδιότητες της κλάσης Model του Flask-SQLAlchemy. Η "μαγική" μεταβλητή __tablename__ ορίζει το όνομα του πίνακα. Η κλάση που ορίσαμε θα έχει πολλαπλά πεδία ως μεταβλητές της. Τα πεδία δημιουργούνται ως *στιγμιότυπα* (*instances*) της κλάσης Column, η οποία λαμβάνει ως παράμετρο τον τύπο του πεδίου καθώς και διάφορες προαιρετικές παραμέτρους, που π.χ. μας επιτρέπουν να υποδείξουμε ποια πεδία θα είναι μοναδικά ή δείκτες. Αναφορικά οι δείκτες σε μια ΒΔ είναι σημαντικοί, καθώς οδηγούν σε γρηγορότερες και συνεπώς αποτελεσματικότερες αναζητήσεις εγγραφών. Σε κάθε χρήστη θα εκχωρηθεί ένα μοναδικό αναγνωριστικό, το οποίο θα είναι το κύριο κλειδί του πίνακα και θα χρησιμεύει στις διαδικασίες ανάγνωσης και ενημέρωσης ή διαγραφής που δείξαμε παραπάνω. Τα αναγνωριστικά αυτά θα είναι τύπου UUID και πιο συγκεκριμένα υπο-τύπου UUID4. Ένα UUID είναι μια ετικέτα 128-bit, η οποία στην πράξη είναι μοναδική ανεξαρτήτως συστήματος. Η πιθανότητα βέβαια να υπάρξει διπλότυπο UUID δεν είναι αποκλειόμενη, αλλά είναι αρκετά κοντά στο μηδέν ώστε να θεωρηθεί αμελητέα. Εφόσον χρησιμοποιούμε την ΒΔ σε διάλεκτο PostgreSQL, θα ορίσουμε τον τύπο ως UUID, κλάση την οποία εισάγουμε από το υποπακέτο dialects.postgresql που υπάγεται στην βιβλιοθήκη SQLAlchemy. Οι πρόσθετες παράμετροι υποδεικνύουν κατά σειρά ότι η στήλη θα είναι το κύριο κλειδί του πίνακα, η προκαθορισμένη τιμή σε περίπτωση που δεν έχει δοθεί θα είναι το αποτέλεσμα της κλήσης της συνάρτησης uuid4 (από το ενσωματωμένο πακέτο uuid), η τιμή πρέπει να είναι μοναδική και όχι κενή. Σημειώνουμε ότι δεν τοποθετήσαμε παρενθέσεις μετά την συνάρτηση uuid4, επομένως μεταβιβάζουμε την ίδια την συνάρτηση και όχι το αποτέλεσμα της κλήσης της, επειδή η SQLAlchemy θα αναλάβει τον ορισμό τιμής στο πεδίο καλώντας αυτήν. Τα πεδία ονόματος χρήστη, ηλεκτρονικής διεύθυνσης, κατακερματισμένου κωδικού και οργανισμού ορίζονται ως τύπου συμβολοσειράς (ισοδύναμο με VARCHAR στην ορολογία της ΒΔ) και καθορίζουμε τα μέγιστα μήκη τους έτσι ώστε η ΒΔ να μπορεί

να βελτιστοποιήσει τη χρήση του χώρου. Επειδή η πλατφόρμα μας οφείλει να υιοθετεί τις βέλτιστες πρακτικές ασφαλείας, δεν θα αποθηκεύουμε τους κωδικούς πρόσβασης των χρηστών όπως δίνονται. Εάν σε κάποια περίπτωση παραβιαστεί η βάση δεδομένων, οι κωδικοί θα είναι προσβάσιμοι και αυτό μπορεί να οδηγήσει σε καταστροφικές συνέπειες για τους χρήστες. Επομένως ο κάθε κωδικός θα υφίσταται *κατακερματισμό (hashing)*, γεγονός που βελτιώνει σημαντικά την ασφάλεια. Οι μεταβλητές-μέλη query_relationship και task_relationship αντικατοπτρίζουν την σχέση του πίνακα με τα μοντέλα SearchQueryModel και TaskModel, τα οποία θα παρατεθούν παρακάτω. Υπενθυμίζουμε ότι ο πίνακας των χρηστών θα έχει τη σχέση ένα προς πολλά με τους δύο αυτούς πίνακες. Για αυτό το λόγο το μοντέλο μας πρέπει να ορίζει τη σχέση ως μεταβλητή με την μέθοδο relationship(). Αυτό δεν είναι ένα πραγματικό πεδίο της ΒΔ, αλλά μια προβολή υψηλού επιπέδου (*high-level view*) της σχέσης μεταξύ των πινάκων. Αυτή η προβολή ορίζεται συνήθως στην πλευρά "ένα" και χρησιμοποιείται ως τρόπος για να αποκτήσουμε πρόσβαση στα "πολλά". Η πρώτη παράμετρος στην μέθοδο relationship() είναι το όνομα της κλάσης που αντιπροσωπεύει την πλευρά "πολλά" της σχέσης. Η πρόσθετη παράμετρος lazy καθορίζει τον τρόπο με τον οποίο φορτώνονται τα σχετικά αντικείμενα κατά την υποβολή ερωτημάτων μέσω σχέσεων. Με την τιμή dynamic δημιουργείται ένα ξεχωριστό SQL ερώτημα για το σχετικό αντικείμενο. Η παράμετρος backref ορίζει το όνομα του χαρακτηριστικού που θα προστεθεί στα αντικείμενα της κλάσης "πολλά", το οποίο θα αναφέρεται πίσω στο αντικείμενο "ένα". Έτσι για παράδειγμα έχοντας ένα αντικείμενο τύπου SearchQuery αποθηκευμένο στην μεταβλητή sq, θα μπορούμε με την έκφραση sq.issuer να ανακτήσουμε ένα αντικείμενο τύπου User.

Κώδικας 4.13: *web/app/models/user.py* - Υλοποίηση μοντέλου ΒΔ χρηστών

```
import uuid
from datetime import datetime, timedelta
from flask import current_app
from flask_login import UserMixin
from sqlalchemy.dialects.postgresql import UUID
from werkzeug.security import generate_password_hash, check_password_hash
from app.db import db
from .token import BlackListTokenModel
from .task import TaskModel as Task

class UserModel(UserMixin, db.Model):
    __tablename__ = 'users'

    id = db.Column(
        UUID(as_uuid=True),
        primary_key=True,
        default=uuid.uuid4,
        unique=True,
        nullable=False
    )
    username = db.Column(db.String(80), unique=True)
    email = db.Column(db.String(120), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    organization = db.Column(db.String(80))
```

```
    query_relationship = db.relationship(
        'SearchQueryModel',
        lazy='dynamic',
        backref='issuer'
    )
    task_relationship = db.relationship(
        'TaskModel',
        lazy='dynamic',
        backref='user'
    )

    def __init__(self, username, email, organization):
        self.username = username
        self.email = email
        self.organization = organization

    def __repr__(self):
        return '<User {}>'.format(self.username)

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
```

Ο δομητής θέτει τις τιμές μόνο για τα πεδία, τα οποία θα εισάγονται από τον χρήστη ως έχουν. Η "μαγική" μέθοδος __repr__() υποδεικνύει στην Python πώς να εκτυπώνει αντικείμενα αυτής της κλάσης, η οποία θα είναι χρήσιμη στην αποσφαλμάτωση. Η μέθοδος set_password() λαμβάνει ως παράμετρο τον κωδικό του χρήστη και τον κατακερματίζει. Ο κατακερματισμός κωδικού πρόσβασης χρησιμοποιείται από το πακέτο Werkzeug, το οποίο είναι ένα από τις βασικές εξαρτήσεις του Flask. Το μόνο που χρειάζεται είναι να περάσουμε τον κωδικό πρόσβασης ως παράμετρο στην υλοποιημένη συνάρτηση generate_password_hash(). Η συνάρτηση αυτή μετατρέπει τον κωδικό πρόσβασης σε μια εκτενή κωδικοποιημένη συμβολοσειρά μέσω μιας σειράς κρυπτογραφικών λειτουργιών, που δεν είναι αμφίδρομες, δηλαδή δεν έχουν αντίστροφη λειτουργία. Έτσι ο κατακερματισμένος κωδικός πρόσβασης δεν μπορεί να χρησιμοποιηθεί για να αποκτηθεί ο πρωτότυπος. Αντίστοιχα, η συνάρτηση check_password() λαμβάνει ένα κατακερματισμένο κωδικό πρόσβασης που έχει δημιουργηθεί προηγουμένως και έναν κωδικό πρόσβασης που εισήγαγε ο χρήστης κατά την στιγμή της προσπάθειας σύνδεσης με την εφαρμογή. Η συνάρτηση επιστρέφει την τιμή της συνάρτησης του Werkzeug check_password_hash(), η οποία θα είναι True εάν ο κωδικός πρόσβασης ταιριάζει με τον κατακερματισμό αλλιώς False.

Κώδικας 4.14: *web/app/models/searchQuery.py* - Υλοποίηση μοντέλου ΒΔ για αναζήτηση δεδομένων

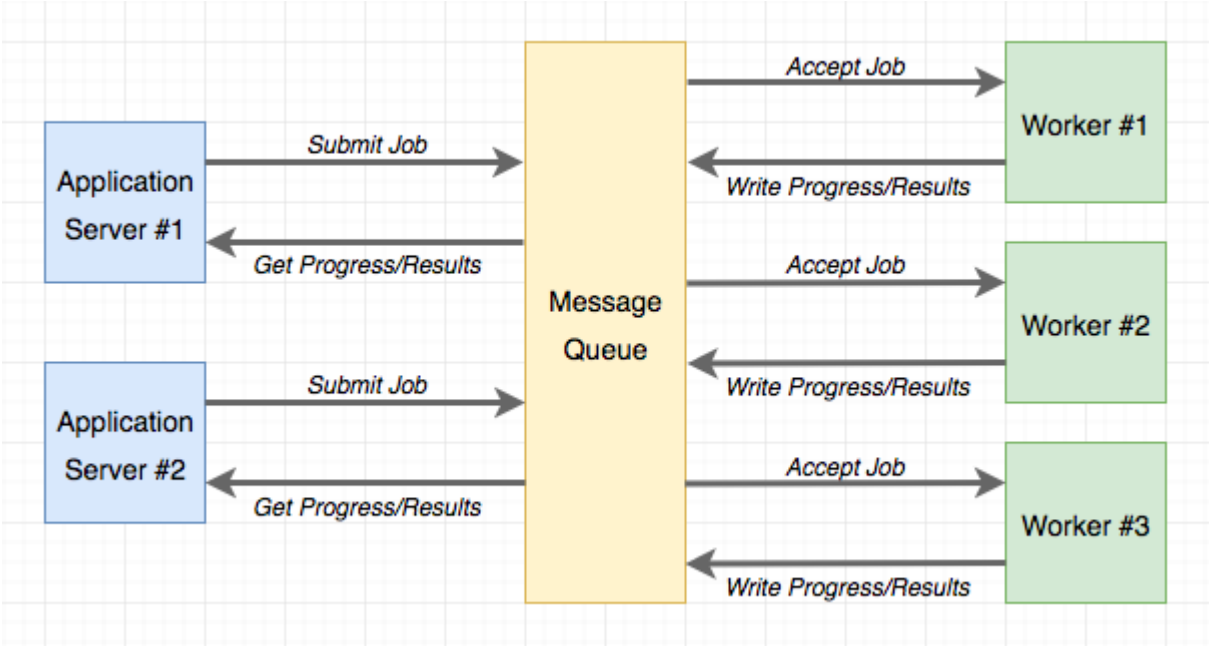
```
from datetime import datetime
from uuid import uuid4
from sqlalchemy.dialects.postgresql import JSON, UUID
from sqlalchemy_utils import ScalarListType
from app.db import db

class SearchQueryModel(db.Model):
      __tablename__ = 'queries'

      id = db.Column(
            UUID(as_uuid=True),
            primary_key=True,
            default=uuid4,
            unique=True,
            nullable=False
      )
      # consent & attributes searched
      data = db.Column(JSON(), nullable=False)
      # Invenio records which fulfill query
      records = db.Column(ScalarListType(), nullable=False)
      # timestamp when the query was made
      submitted_at = db.Column(
            db.DateTime(timezone=True),
            default=datetime.utcnow,
            nullable=False
      )
      user_id = db.Column(
UUID(), db.ForeignKey('users.id'), nullable=False)
```

Ο επόμενος πίνακας που θα υλοποιήσουμε είναι τα ερωτήματα αναζήτησης δεδομένων από τους αναλυτές για την υλοποίηση πειραμάτων. Τα ερωτήματα αυτά πρέπει να πληρούν την συναίνεση που έχει δώσει ο κάθε χρήστης. Ακόμη το ερώτημα θα πρέπει να ταιριάζει με εκείνα τα δεδομένα, στα οποία οι ιδιότητες προς ανάλυση είναι υπάρχουσες. Για παράδειγμα εάν ο αναλυτής επιθυμεί να πραγματοποιήσει πείραμα συσχέτισης ηλικίας και μέτρησης πίεσης αίματος, δεν θα πρέπει να ανακτώνται δεδομένα τα οποία δεν περιλαμβάνουν αυτά τα χαρακτηριστικά ακόμα και αν πληρείται η συναίνεση. Στον κώδικα 4.14 παρουσιάζουμε την υλοποίηση του μοντέλου αυτού.

Κάθε ερώτημα αναζήτησης είναι μοναδικό και αντιπροσωπεύεται επίσης με ένα μοναδικό αναγνωριστικό τύπου uuid4, το οποίο θα είναι και το κύριο κλειδί του πίνακα. Το επόμενο πεδίο είναι τα δεδομένα που έχουν ζητηθεί, τα οποία είναι στην ουσία τα πεδία συναίνεσης και τα χαρακτηριστικά προς χρήση για το πείραμα. Οι τιμές της στήλης αυτής θα είναι τύπου JSON, ένας ειδικός τύπος στήλης που υποστηρίζεται από την PostgreSQL. Με αυτόν τον τύπο αποφεύγουμε τη δημιουργία διαφορετικής στήλης για κάθε υποκατηγορία δεδομένων. Επίσης τα δεδομένα προς αναζήτηση θα εισέρχονται στην εφαρμογή σε μορφή λεξικών, γεγονός που διευκολύνει ακόμα περισσότερο την αποθήκευση και

ανάκτησή τους από την ΒΔ. Η επόμενη στήλη, δηλωμένη ως πεδίο της κλάσης με το όνομα records, αποθηκεύει τις τιμές όλων των Persistent Identifiers που χρησιμοποιεί το ασφαλές αποθετήριο δεδομένων, των οποίων τα μεταδεδομένα πληρούν τις τιμές αναζήτησης. Η κατηγορία της στήλης είναι μια ειδική περίπτωση που ονομάζεται ScalarListType, η οποία παρέχεται από το πακέτο sqlalchemy_utils. Ο τύπος αυτός παρέχει τρόπο αποθήκευσης πολλαπλών βαθμωτών τιμών σε μία στήλη πίνακα ΒΔ. Λειτουργεί ως λίστα από την πλευρά της Python και αποθηκεύει το αποτέλεσμα ως λίστα με τιμές διαχωρισμένες με κόμμα στην ΒΔ (ο τύπος στην ΒΔ θα είναι TEXT). Υπενθυμίζουμε πως οι Persistent Identifiers είναι ακέραιοι, οπότε για παράδειγμα η λίστα θα έχει τη μορφή [1, 2, 3, …] στην Python και 1,2,3,.. στην ΒΔ. Χρησιμοποιώντας τη μορφή λίστας μπορούμε να παρουσιάσουμε εύκολα το πλήθος των δεδομένων που πληρούν το εκάστοτε ερώτημα αναζήτησης με την έκφραση length(records). Η στήλη submitted_at είναι τύπου DateTime και η τιμή της θα είναι μια χρονική σήμανση, η οποία είναι χρήσιμη για τις καταγραφές προέλευσης. Στον δομητή τύπου DateTime έχουμε ορίσει την παράμετρο timezone ως αληθή, γεγονός που υποδεικνύει στην ΒΔ να αποθηκεύει τις χρονικές σημάνσεις με ζώνη ώρας. Γενικά, η προτιμώμενη τεχνική είναι να εργαζόμαστε με ημερομηνίες και ώρες UTC. Αυτό διασφαλίζει ότι χρησιμοποιούμε ομοιόμορφες χρονικές σημάνσεις, ανεξαρτήτως από το που βρίσκεται ο χρήστης. Αυτές οι χρονικές σημάνσεις μπορούν να μετατραπούν στην τοπική ώρα του χρήστη κατά την παρουσίασή τους. Η προεπιλεγμένη τιμή θα συμπληρώνεται από την εφαρμογή και για αυτό το λόγο περνάμε ως τιμή της παραμέτρου default τη συνάρτηση datetime.utcnow. Το πεδίο user_id αρχικοποιείται ως ξένο κλειδί στην έκφραση users.id, που σημαίνει ότι αναφέρεται στην τιμή του μοναδικού αναγνωριστικού του χρήστη από τον πίνακα users. Τονίζουμε ότι στην περίπτωση χρήσης της κλάσης ForeignKey() το όνομα του μοντέλου που χρησιμοποιούμε είναι αυτό της τιμής του μέλους-μεταβλητής __tablename__ (δηλαδή το όνομα του πίνακα στη ΒΔ) και όχι το όνομα της κλάσης.

Τέλος υλοποιούμε το μοντέλο ΒΔ TaskModel, το οποίο αποθηκεύει πληροφορίες σχετικές με τις *ουρές εργασιών (task queues)*. Οι ουρές εργασιών παρέχουν λύση για τις εφαρμογές να ζητούν εκτέλεση μιας εργασίας μέσω μιας *διεργασίας worker (worker process)*. Οι διεργασίες worker εκτελούνται ανεξάρτητα από την εφαρμογή και μπορούν ακόμη και να εντοπιστούν σε διαφορετικό σύστημα. Η επικοινωνία μεταξύ της εφαρμογής και των worker γίνεται μέσω μιας *ουράς μηνυμάτων (message queue)*. Η εφαρμογή υποβάλλει μια εργασία και στη συνέχεια, παρακολουθεί την πρόοδό της αλληλεπιδρώντας με την ουρά. Το σχήμα 4.2 παρουσιάζει μια τέτοια τυπική υλοποίηση. Σε αυτό το σημείο κρίνεται αναγκαίο να αναπαραστήσουμε την ουρά εργασιών ως μοντέλο ΒΔ επειδή στις εφαρμογές ιστού, μόλις ξεκινήσει μια εργασία ως μέρος ενός αιτήματος HTTP, το αίτημα αυτό θα τερματιστεί και όλο το πλαίσιο για την εργασία αυτή θα χαθεί. Για να διατηρήσουμε λοιπόν κάποια κατάσταση της εργασίας πρέπει να χρησιμοποιήσουμε έναν πίνακα, επειδή η εφαρμογή πρέπει να παρακολουθεί ποιες εργασίες εκτελεί κάθε χρήστης.



Σχήμα 4.2: Παράδειγμα ουράς εργασιών

Στον κώδικα 4.15 παρουσιάζουμε την υλοποίηση του μοντέλου των εργασιών. Το μοντέλο θα αποθηκεύει το μοναδικό αναγνωριστικό τύπου UUID4 της εργασίας, το πλήρως αναγνωρισμένο όνομά της (όπως αυτό μεταβιβάστηκε στην RQ), μια περιγραφή για την εργασία που είναι κατάλληλη για εμφάνιση στους χρήστες, το όνομα του καταλόγου στο οποίο θα εξάγονται αρχεία (εάν χρειάζεται) το αναγνωριστικό του χρήστη που ζήτησε την εργασία και μία boolean τιμή που υποδεικνύει εάν η εργασία ολοκληρώθηκε ή όχι. Ο σκοπός του πεδίου complete είναι να διαχωρίσει τις εργασίες που έληξαν από αυτές που εκτελούνται ενεργά, καθώς οι εργασίες που εκτελούνται απαιτούν ειδικό χειρισμό για την εμφάνιση ενημερώσεων προόδου.

Κώδικας 4.15: *web/app/models/task.py* - Υλοποίηση μοντέλου ΒΔ για ασύγχρονες εργασίες

```
from uuid import uuid4
from flask import current_app
from redis.exceptions import RedisError
from rq.exceptions import NoSuchJobError
from rq.job import Job
from sqlalchemy.dialects.postgresql import UUID
from app.db import db

class TaskModel(db.Model):
      __tablename__ = 'tasks'

      id = db.Column(
            UUID(as_uuid=True),
            primary_key=True,
            default=uuid4,
            unique=True,
```

```
            nullable=False
      )
      name = db.Column(db.String(128), index=True)
      description = db.Column(db.String(128))
      directory = db.Column(db.String(80), default=None)
      user_id = db.Column(
            UUID(as_uuid=True),
            db.ForeignKey('users.id'),
            nullable=False
      )
      complete = db.Column(db.Boolean, default=False)

      def get_job(self):
            try:
                  rq_job = Job.fetch(
                        str(self.id), connection=current_app.redis
                  )
            except (RedisError, NoSuchJobError):
                  return None
            return rq_job

      def get_progress(self):
            job = self.get_job()
            return job.meta.get('progress', 0) if job is not None else
100

      def to_dict(self):
            return dict(
                  id=str(self.id),
                  name=self.name,
                  description=self.description,
                  progress=self.get_progress(),
                  complete=self.complete
            )
```

Η μέθοδος get_rq_job() είναι μια βοηθητική μέθοδος που φορτώνει το στιγμιότυπο RQ Job, δεδομένου ενός αναγνωριστικού εργασίας, το οποίο μπορούμε να λάβουμε από το μοντέλο. Αυτό γίνεται με το Job.fetch(), το οποίο φορτώνει το στιγμιότυπο Job από τα δεδομένα που υπάρχουν στο Redis σχετικά με αυτό. Η μέθοδος get_progress() βασίζεται στη μέθοδο get_job() και επιστρέφει το ποσοστό προόδου για την εργασία. Εάν το αναγνωριστικό εργασίας από το μοντέλο δεν υπάρχει στην ουρά RQ, αυτό σημαίνει ότι η εργασία έχει ήδη ολοκληρωθεί και τα δεδομένα έχουν λήξει και αφαιρεθεί από την ουρά, οπότε στην περίπτωση αυτή το ποσοστό που επιστρέφεται είναι 100. Από την άλλη μεριά, εάν η εργασία υπάρχει, αλλά δεν υπάρχουν πληροφορίες που να σχετίζονται με το χαρακτηριστικό meta, τότε είναι ασφαλές να υποθέσουμε ότι η εργασία έχει προγραμματιστεί να εκτελεστεί, αλλά δεν είχε ακόμη την ευκαιρία να ξεκινήσει, οπότε σε αυτήν την περίπτωση το 0 επιστρέφεται ως πρόοδος.

4.2.2.6 Λειτουργίες Διεπαφής Χρήστη

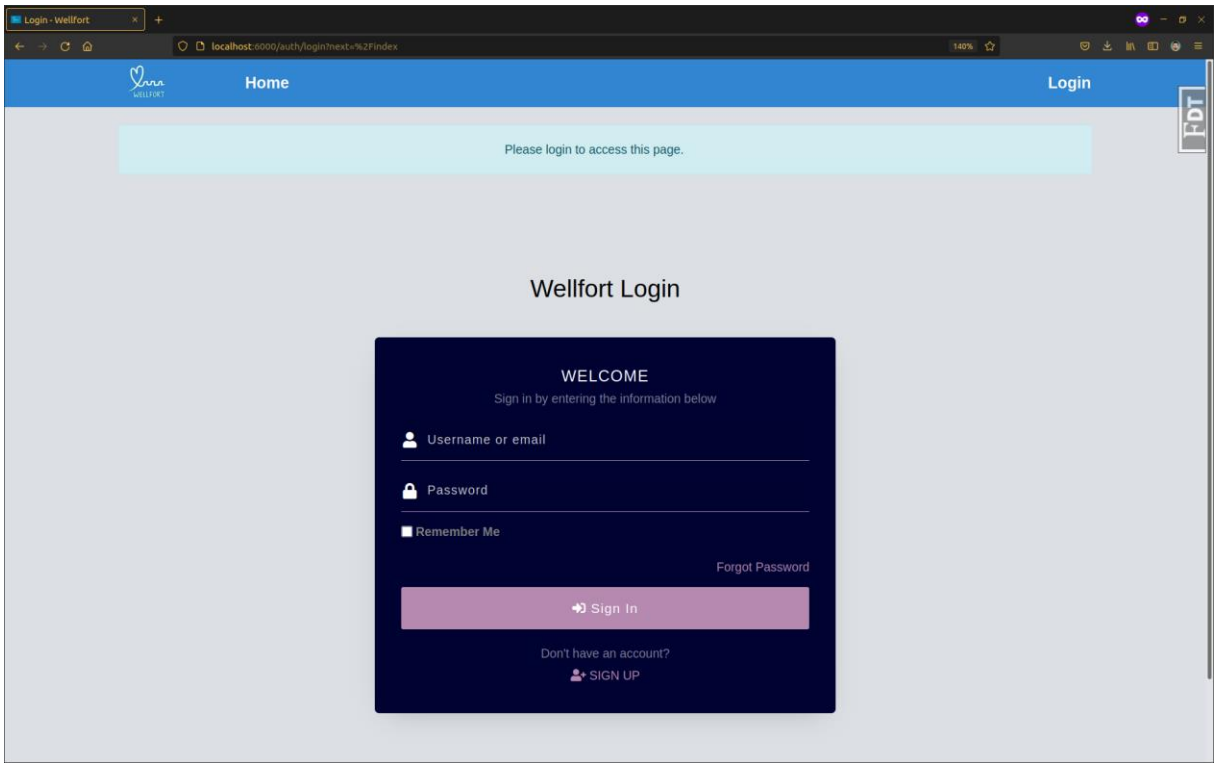
Η υλοποίηση της ΔΧ του πρωτοτύπου παρέχει τις βασικές λειτουργίες ασφαλείας όπως έλεγχο ταυτότητας χρηστών και τρεις βασικές μεθόδους ανάλυσης δεδομένων, με σεβασμό πάντα στο απόρρητο: ανάλυση, σύνθεση και ανωνυμοποίηση δεδομένων.

4.2.2.6.1 Έλεγχος ταυτότητας χρηστών

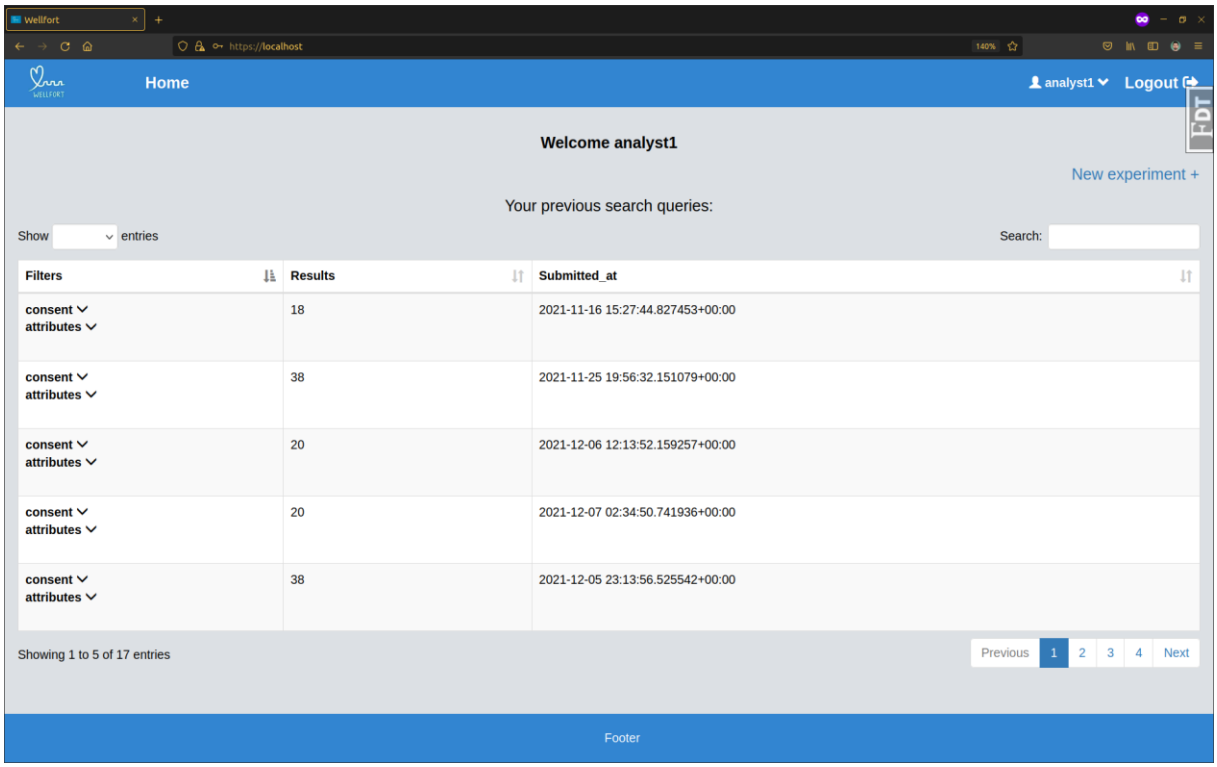
Για λόγους ασφαλείας, οι περισσότερες διαδρομές της ΔΧ είναι προστατευμένες και ο χρήστης θα πρέπει να ταυτοποιηθεί ώστε να έχει πρόσβαση σε αυτές. Στη σελίδα αυτή ο χρήστης πρέπει να εισάγει όνομα χρήστη ή ηλεκτρονικής διεύθυνσης ταχυδρομείου και κωδικό (εφόσον φυσικά έχει εγγραφεί στην πλατφόρμα). Η εικόνα 4.2 παρουσιάζει την σελίδα login για τον αναλυτή. Εφόσον ο αναλυτής παρέχει σωστά credentials, τότε θα ανακατευθυνθεί στην αρχική σελίδα, όπου και παρουσιάζεται μια επισκόπηση των εναπομείναντων και ολοκληρωμένων ερωτημάτων που έχει πραγματοποιήσει. Για λόγους περαιτέρω ασφαλείας θα ορίσουμε και ένα χρονικό όριο συνεδρίας (session lifetime). Εάν αυτό το χρονικό διάστημα παρέλθει, ο χρήστης θα πρέπει να συνδεθεί εκ νέου στην πλατφόρμα.

Στην εικόνα 4.3 αποτυπώνεται ένα στιγμιότυπο της αρχικής σελίδας, όπου ο πίνακας περιέχει μια σύνοψη των προηγούμενων ερωτημάτων αναζήτησης του αναλυτή. Σε αυτόν τον πίνακα ο αναλυτής μπορεί να συνεχίσει ερωτήματα τα οποία είναι εκκρεμή από προηγούμενη αναζήτηση, ώστε να μην χρειάζεται να επαναλαμβάνει την καταχώριση φίλτρων.

Η εμφάνιση του πίνακα γίνεται με την χρήση της βιβλιοθήκης dataTables.js, η οποία αναλαμβάνει την δημιουργία πινάκων και την προσθήκη αλληλεπιδράσεων σε αυτούς. Παρέχει αναζήτηση, ταξινόμηση και σελιδοποίηση χωρίς περαιτέρω διαμόρφωση.



Εικόνα 4.2: Απεικόνιση ελέγχου ταυτότητας χρήστη



Εικόνα 4.3: Απεικόνιση αρχικής σελίδας ταυτοποιημένου χρήστη

Όπως προαναφέραμε, για τις λειτουργίες login στη ΔΧ, χρησιμοποιούμε λειτουργίες και της επέκτασης flask-login. Η επέκταση αυτή μας παρέχει το αντικείμενο current_user, το οποίο μπορεί να χρησιμοποιηθεί ανά πάσα στιγμή κατά τη διάρκεια του χειρισμού ενός αιτήματος, για να ληφθεί το αντικείμενο χρήστη που αντιπροσωπεύει τον πελάτη του αιτήματος. Ακόμη οι υλοποιημένες συναρτήσεις login_user() και logout_user() είναι βολικές για είσοδο και έξοδο του χρήστη αντίστοιχα. Να επισημάνουμε πως θα χρειαστεί να παράσχουμε μια *επιστροφή κλήσης (callback)* ονόματι user_loader. Αυτή η επανάκληση χρησιμοποιείται για τη φόρτωση εκ νέου του αντικειμένου χρήστη από το αναγνωριστικό του, που είναι αποθηκευμένο στη συνεδρία. Λαμβάνει το αναγνωριστικό unicode και έτσι επιστρέφει το αντίστοιχο αντικείμενο χρήστη. Εάν ο χρήστης έχει ταυτοποιηθεί, το αντικείμενο αυτό θα δείχνει σε αντικείμενο χρήστη από τη ΒΔ (όπως δείξαμε στον κώδικα 4.13), εάν όχι, σε ένα ειδικό ανώνυμο αντικείμενο χρήστη.

Κώδικας 4.16: *web/app/auth/routes.py* - Υλοποίηση ταυτοποίησης χρήστη

```
from re import fullmatch
from flask import current_app, flash, redirect, render_template, request,
session, url_for
from flask_login import current_user, login_user
from werkzeug.urls import url_parse
from app.auth import auth_bp

@auth_bp.route('/login', methods=('GET', 'POST'))
def login():
    if current_user.is_authenticated:
```

```
        return redirect(url_for('index'))
    form = LoginForm()
    if request.method == 'POST' and form.validate_on_submit():
        username_or_email = form.username.data
        regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
        email = True if fullmatch(regex, username_or_email) else False
        if not email:
            user =
User.query.filter_by(username=username_or_email).first()
        else:
            user = User.query.filter_by(email=username_or_email).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password', 'danger')
            return redirect(url_for('auth.login'))
        login_user(user, remember=form.remember_me.data)
        session['token'] =
user.encode_token(current_app.permanent_session_lifetime).decode()

        next_page = request.args.get('next')
        if not next_page or url_parse(next_page).netloc != '':
            next_page = url_for('index')

        flash('Successfully logged in.', 'success')
        return redirect(next_page)

    return render_template('auth/login.html', title='Login', form=form)
```

Ο κώδικας 4.16 απεικονίζει την βασική λειτουργία login της εφαρμογής μας. Στον διακοσμητή δηλώνουμε την διαδρομή και τις επιτρεπόμενες HTTP μεθόδους για αυτήν την διαδρομή. Οι δύο πρώτες γραμμές χειρίζονται την περίπτωση όπου ο χρήστης είναι συνδεδεμένος και πλοηγείται σε αυτήν την διαδρομή. Εδώ γίνεται σαφές, ότι αυτό δεν πρέπει να επιτραπεί. Επομένως όταν ο χρήστης είναι ήδη συνδεδεμένος, θα ανακατευθύνεται στην αρχική σελίδα.

Η μεταβλητή form είναι ένα αντικείμενο τύπου LoginForm(), που έχουμε ορίσει στο module forms.py. Συνοπτικά, η κλάση αυτή κληρονομεί χαρακτηριστικά από την κλάση FlaskForm του python πακέτου flask-wtf, το οποίο μας επιτρέπει να δηλώνουμε διάφορες φόρμες ως κλάσεις Python. Μια κλάση φόρμας ορίζει τα πεδία της ως μεταβλητές κλάσης.

Ακολούθως ελέγχουμε την μέθοδο HTTP που έχει αποσταλεί. Εάν η μέθοδος είναι GET, τότε πιθανώς έχει ζητηθεί η σελίδα login για την εισαγωγή στοιχείων στη φόρμα, όπου και επιστρέφουμε την HTML σελίδα, η οποία αναλαμβάνει την δημιουργία της φόρμας. Εάν η μέθοδος είναι POST, αυτό σημαίνει ότι η φόρμα έχει υποβληθεί, όπου και πρέπει να διαβαστούν τα συμπληρωθέντα στοιχεία της φόρμας. Υπογραμμίζουμε ότι ο έλεγχος αυτός χρησιμοποιείται με σύζευξη με την μέθοδο validate_on_submit(), για έλεγχο πληρότητας των περιορισμών των στοιχείων της φόρμας. Για παράδειγμα, η φόρμα μπορεί να έχει υποχρεωτικά πεδία (όπως σε αυτήν την περίπτωση, καθώς ο χρήστης πρέπει να συμπληρώσει τουλάχιστον όνομα χρήστη ή ηλεκτρονική διεύθυνση ταχυδρομείου και κωδικό πρόσβασης),

περιορισμένο μήκος χαρακτήρων ή αριθμητικό τύπο ορισμένου εύρους για κάποια πεδία. Με την έκφραση form.username.data λαμβάνουμε την τιμή που εισήχθη στο πεδίο username και την αναθέτουμε στην μεταβλητή username_or_email, επειδή η αλφαριθμητική τιμή μπορεί να είναι είτε όνομα χρήστη είτε διεύθυνση ηλεκτρονικού ταχυδρομείου.

Κατόπιν θέτουμε μια μεταβλητή email, η οποία θα είναι αληθής αν ο χρήστης έχει εισάγει μοτίβο που παραπέμπει σε έγκυρη διεύθυνση ηλεκτρονικού ταχυδρομείου, ψευδής στην αντίθετη περίπτωση. Ο έλεγχος για μοτίβο ηλεκτρονικού ταχυδρομείου γίνεται με την χρήση *κανονικής έκφρασης (regular expression)*, την οποία περνάμε ως παράμετρο στην συνάρτηση full_match() της ενσωματωμένης βιβλιοθήκης "re". Εάν η συνάρτηση ανιχνεύσει αυτό το μοτίβο (το οποίο ορίσαμε ως κανονική έκφραση στην μεταβλητή regex), τότε ψάχνουμε την ύπαρξη της δοθείσας διεύθυνσης ταχυδρομείου στη ΒΔ, αλλιώς πραγματοποιούμε αναζήτηση με βάση το όνομα χρήστη. Το ερώτημα ΒΔ που χρησιμοποιούμε είναι πανομοιότυπο με αυτό που παραθέσαμε στον κώδικα 4.12. Δεδομένου ότι γνωρίζουμε πως θα υπάρχουν ακριβώς μόνο ένα ή μηδέν αποτελέσματα, ολοκληρώνουμε το ερώτημα καλώντας τη συνάρτηση first(), η οποία θα επιστρέψει το αντικείμενο χρήστη εάν υπάρχει ή None εάν δεν υπάρχει.

Εάν λάβουμε αντιστοιχία για το όνομα χρήστη ή τη διεύθυνση ηλεκτρονικού ταχυδρομείου που δόθηκε, μπορούμε στη συνέχεια να ελέγξουμε εάν ο κωδικός πρόσβασης που συνοδεύει τη φόρμα είναι έγκυρος. Αυτό γίνεται με την κλήση της μεθόδου check_password() που ορίσαμε στο μοντέλο ΒΔ χρήστη (Κώδικας 4.13). Η μέθοδος θα λάβει τον κατακερματισμό του κωδικού πρόσβασης που είναι αποθηκευμένος για τον εκάστοτε χρήστη και θα καθορίσει εάν ο κωδικός πρόσβασης που έχει εισαχθεί στη φόρμα ταιριάζει με τον κατακερματισμό ή όχι. Τώρα λοιπόν έχουμε δύο πιθανές συνθήκες σφάλματος: το όνομα χρήστη ή η διεύθυνση ηλεκτρονικού ταχυδρομείου μπορεί να μην είναι έγκυρα ή ο κωδικός πρόσβασης μπορεί να είναι λανθασμένος για τον χρήστη. Σε οποιαδήποτε από αυτές τις περιπτώσεις, *αναβοσβήνουμε (flash)* ένα μήνυμα για την αποτυχία ταυτοποίησης και ανακατευθύνουμε πίσω στη σελίδα login, ώστε ο χρήστης να προσπαθήσει ξανά. Το *σύστημα flashing (flashing system)* καθιστά δυνατή την εγγραφή ενός μηνύματος στο τέλος ενός HTTP αιτήματος και την πρόσβαση σε αυτό μόνο στο επόμενο αίτημα. Σημειώνουμε ότι στην συνάρτηση url_for(), η οποία παράγει τους ΕΕΠ, περνάμε ως συμβολοσειρά το όνομα του σχεδιαγράμματος και το όνομα της συνάρτησης που ανήκει σε αυτό, διαχωρισμένα από μία τελεία.

Στην περίπτωση που το όνομα χρήστη ή η ΔΗΤ και ο κωδικός πρόσβασης είναι σωστά, τότε καλούμε την συνάρτηση login_user(), η οποία προέρχεται από το πακέτο Flask-Login. Αυτή η συνάρτηση θα εγγράψει τον χρήστη ως συνδεδεμένο, γεγονός που σημαίνει ότι σε όποιες μελλοντικές σελίδες πλοηγηθεί, η μεταβλητή current_user σε αυτές τις σελίδες θα "δείχνει" ή καλύτερα θα οριστεί στον συγκεκριμένο χρήστη.

Η μεταβλητή next_page λαμβάνει την τιμή της παραμέτρου next από τον ΕΕΠ. Το Flask παρέχει μια μεταβλητή request, που περιέχει όλες τις πληροφορίες που έστειλε ο πελάτης με το αίτημα. Γενικά, όταν ο χρήστης δεν είναι ταυτοποιημένος και μεταβαίνει σε προστατευμένες διαδρομές (αυτές με το διακοσμητή @login_required), ο διακοσμητής θα ανακατευθύνει στη σελίδα σύνδεσης, αλλά θα συμπεριλάβει κάποιες επιπλέον πληροφορίες σε αυτήν την ανακατεύθυνση, ώστε η εφαρμογή να μπορεί στη συνέχεια να επιστρέψει στην πρώτη σελίδα. Στην περίπτωσή μας, θα παρεμποδίσει το αίτημα και θα απαντήσει με μια ανακατεύθυνση στη σελίδα login (/auth/login). Εκτός αυτού, θα προσθέσει και ένα όρισμα συμβολοσειράς ερωτήματος σε αυτόν τον ΕΕΠ, δημιουργώντας τον πλήρες ΕΕΠ ανακατεύθυνσης ως /auth/login?next=/. Το όρισμα next εμπεριέχει την τιμή του αρχικού ΕΕΠ που ζητήθηκε, ώστε η εφαρμογή να μπορεί να την χρησιμοποιήσει για να ανακατευθύνει πίσω σε αυτόν,

κατόπιν επιτυχούς ταυτοποίησης. Για αυτόν λοιπόν το λόγο, αποθηκεύουμε την τιμή στη μεταβλητή next_page και στη συνθήκη if, ελέγχουμε τα εξής: i) Εάν ο ΕΕΠ σύνδεσης δεν έχει την παράμετρο next ορισμένη, τότε ο χρήστης ανακατευθύνεται στην αρχική σελίδα. ii) Εάν η παράμετρος αυτή περιέχει τιμή που δείχνει σε μια πλήρη διεύθυνση ΕΕΠ που περιλαμβάνει όνομα τομέα, τότε ο χρήστης επίσης ανακατευθύνεται στην αρχική σελίδα, για λόγους ασφαλείας. iii) Εάν δεν ικανοποιείται καμία συνθήκη από τις παραπάνω, η τιμή θα έχει οριστεί σε μια σχετική διαδρομή (με άλλα λόγια, μια διεύθυνση ΕΕΠ χωρίς το τμήμα τομέα) και τότε ο χρήστης ανακατευθύνεται σε αυτήν. Για να προσδιορίσουμε εάν ο ΕΕΠ είναι σχετικός ή απόλυτος, τον αναλύουμε με τη συνάρτηση url_parse() του Werkzeug και, στη συνέχεια, ελέγχουμε εάν το στοιχείο netloc έχει οριστεί ή όχι.

4.2.2.6.2 Αναζήτηση δεδομένων

4.2.2.6.2.1 Εφαρμογή φίλτρων

Η αναζήτηση δεδομένων προϋποθέτει την επιτυχή ταυτοποίηση του αναλυτή με την πλατφόρμα. Εάν ο αναλυτής μεταβεί στο σύνδεσμο New Experiment στην αρχική σελίδα, τότε ανακατευθύνεται στην φόρμα συγκατάθεσης προς αναζήτηση δεδομένων. Η φόρμα αυτή παρουσιάζεται στην εικόνα 4.4 και περιέχει 5 στοιχεία τα οποία είναι όλα υποχρεωτικά.

- Το πρώτο πεδίο της φόρμας είναι ο οργανισμός στον οποίο ανήκει ο αναλυτής και είναι μόνο προς ανάγνωση. Η τιμή του ανακτάται από την ΒΔ. Ο αναλυτής ανήκει στην εταιρία Α, οπότε το φίλτρο εφαρμόζεται αυτόματα στην αναζήτηση δεδομένων.
- Το πεδίο Purpose αντιπροσωπεύει το σκοπό του πειράματος, όπου και η τιμή θα ταιριάζει μόνο σε δεδομένα, για τα οποία ο σκοπός έχει την ίδια τιμή με αυτήν που δοθεί από τον αναλυτή. Μερικά παραδείγματα τιμών αποτελούν οι Commercial Interest (εμπορικό ενδιαφέρον), Research And Development (έρευνα και ανάπτυξη), Security (ασφάλεια). Εδώ επιλέγουμε την τιμή Research And Development.
- Στο πεδίο Description πρέπει να εισαχθεί μια απλή περιγραφή, η οποία θα πρέπει να περιέχει μια περίληψη για το πείραμα και τον στόχο του.
- Το πεδίο Processing αντιπροσωπεύει την ενέργεια που θα εκτελεστεί στα δεδομένα. Για παράδειγμα, εάν επιλεχθεί η ανάλυση δεδομένων, τότε η πλατφόρμα θα ξεκινήσει το περιβάλλον ανάλυσης μετά από επιβεβαίωση. Μερικές τιμές που ανήκουν σε αυτήν την κατηγορία είναι οι Adapt (προσαρμογή), Analyse (ανάλυση), Anonymise (ανωνυμοποίηση), Combine (συνδυασμός), Share (διαμοιρασμός). Θα προχωρήσουμε σε ανάλυση των δεδομένων χρησιμοποιώντας το Ασφαλές Περιβάλλον Ανάλυσης, συνεπώς επιλέγουμε και την τιμή Analyse.
- Τέλος, το πεδίο Duration (διάρκεια) είναι η ημερομηνία λήξης της μελέτης και θα αντιστοιχίσει δεδομένα για τα οποία η ημερομηνία λήξης της συγκατάθεσης τους έχει οριστεί σε μεταγενέστερη ημερομηνία από την επιλεγμένη.

Κατόπιν της υποβολής της φόρμας, οι επιλεχθέντες τιμές αποθηκεύονται προσωρινά στη συνεδρία του αναλυτή, ώστε να διατηρηθούν μεταξύ των αιτημάτων HTTP. Στη συνέχεια ο αναλυτής καλείται να συμπληρώσει την φόρμα χαρακτηριστικών, τις οποίες θα χρησιμοποιήσει στο πείραμα του. Φυσικά πρέπει να επιλεγεί τουλάχιστον ένα checkbox από οποιαδήποτε κατηγορία. Η φόρμα παρουσιάζει τα χαρακτηριστικά οργανωμένα σε δύο μεγάλες κατηγορίες, Patient (Ασθενή) και Observation (Παρατήρηση). Κάθε μία περιέχει υποκατηγορίες, σχετικές με τις γονικές κατηγορίες. Κάθε υποκατηγορία περιέχει με τη σειρά της μια ομάδα χαρακτηριστικών ως checkboxes που συσχετίζονται φυσικά με τη μητρική υποκατηγορία.

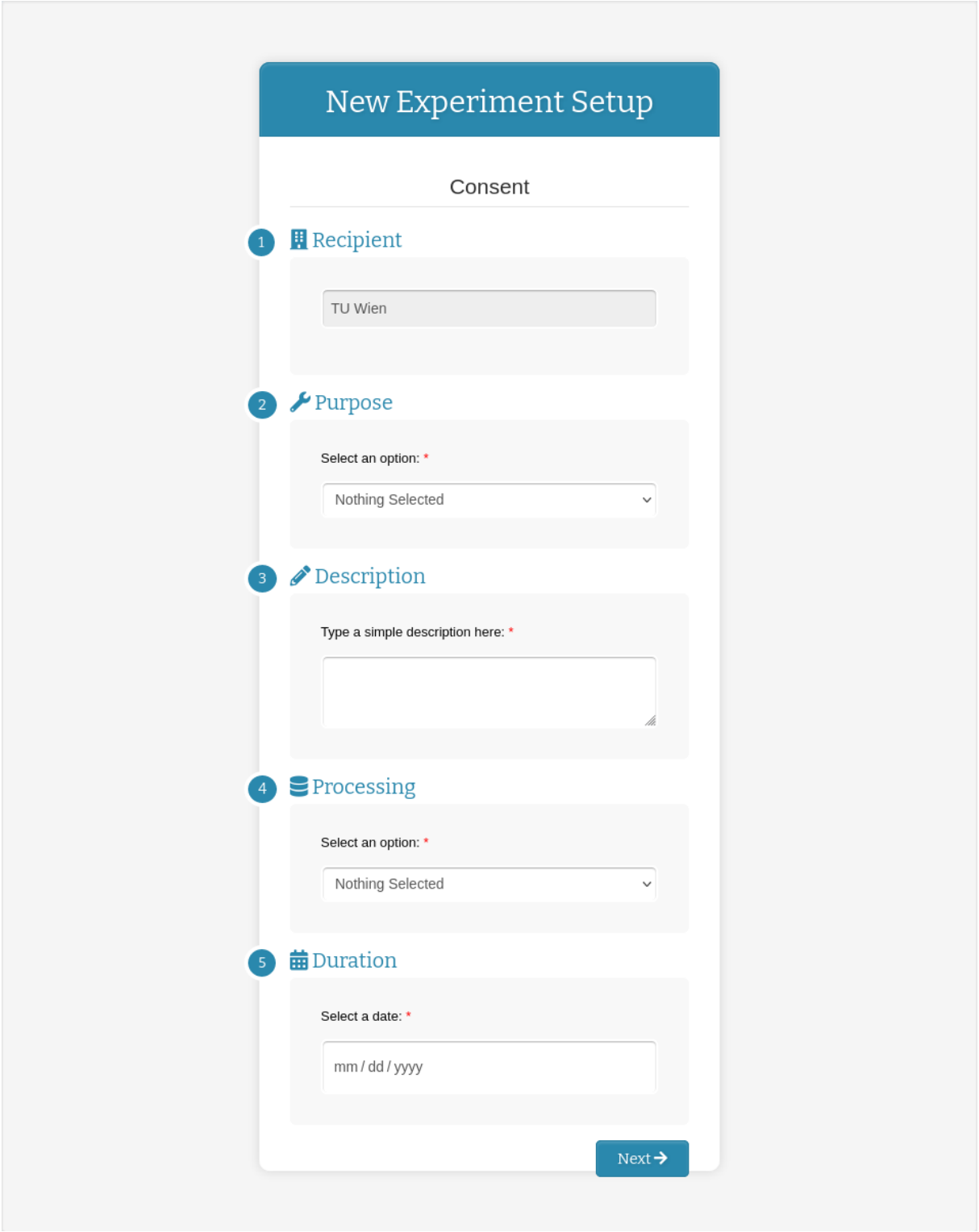
Η επιλογή χαρακτηριστικών είναι διττή: Κάθε επιλεγμένο χαρακτηριστικό: i) εφαρμόζεται ως φίλτρο κατά την ανάκτηση των συνόλων δεδομένων ii) θα εξαχθεί στη λειτουργία αποτελέσματος (σε αυτήν την περίπτωση στον διακομιστή ανάλυσης). Κάθε υποκατηγορία είναι επεκτάσιμη και εμφανίζει τα χαρακτηριστικά ως checkboxes κάνοντας κλικ. Στην εικόνα 4.5 παρακάτω, τα χαρακτηριστικά Gender (Φύλο), Age (Ηλικία) και Pulse Respiration Quotient (Πηλίκο αναπνευστικού-καρδιακού ρυθμού) επιλέγονται για εξαγωγή.

Η αναζήτηση θα είναι ουσιαστικά ένα ερώτημα προς την υπηρεσία Elasticsearch, το οποίο θα περιέχει τα φίλτρα συναίνεσης και χαρακτηριστικών. Για να είναι περισσότερο κατανοητή η υλοποίηση, την απομονώνουμε σε ξεχωριστό module. Η υλοποίηση της αναζήτησης παρουσιάζεται στον κώδικα 4.17. Η συνάρτηση query_index() λαμβάνει ως υποχρεωτικές παραμέτρους το όνομα ευρετηρίου και τα φίλτρα (σε μορφή λεξικού) που θα εφαρμοστούν. Οι παράμετροι from, size και source είναι προαιρετικές με προκαθορισμένες τιμές 1, 10 και False αντίστοιχα. Οι δύο πρώτες θα ελέγχουν ποιο υποσύνολο ολόκληρου του συνόλου αποτελεσμάτων πρέπει να επιστραφεί. Η παράμετρος source υποδεικνύει εάν θα επιστραφούν τα μεταδεδομένα της εκάστοτε αναζήτησης (θα αναλυθεί παρακάτω).

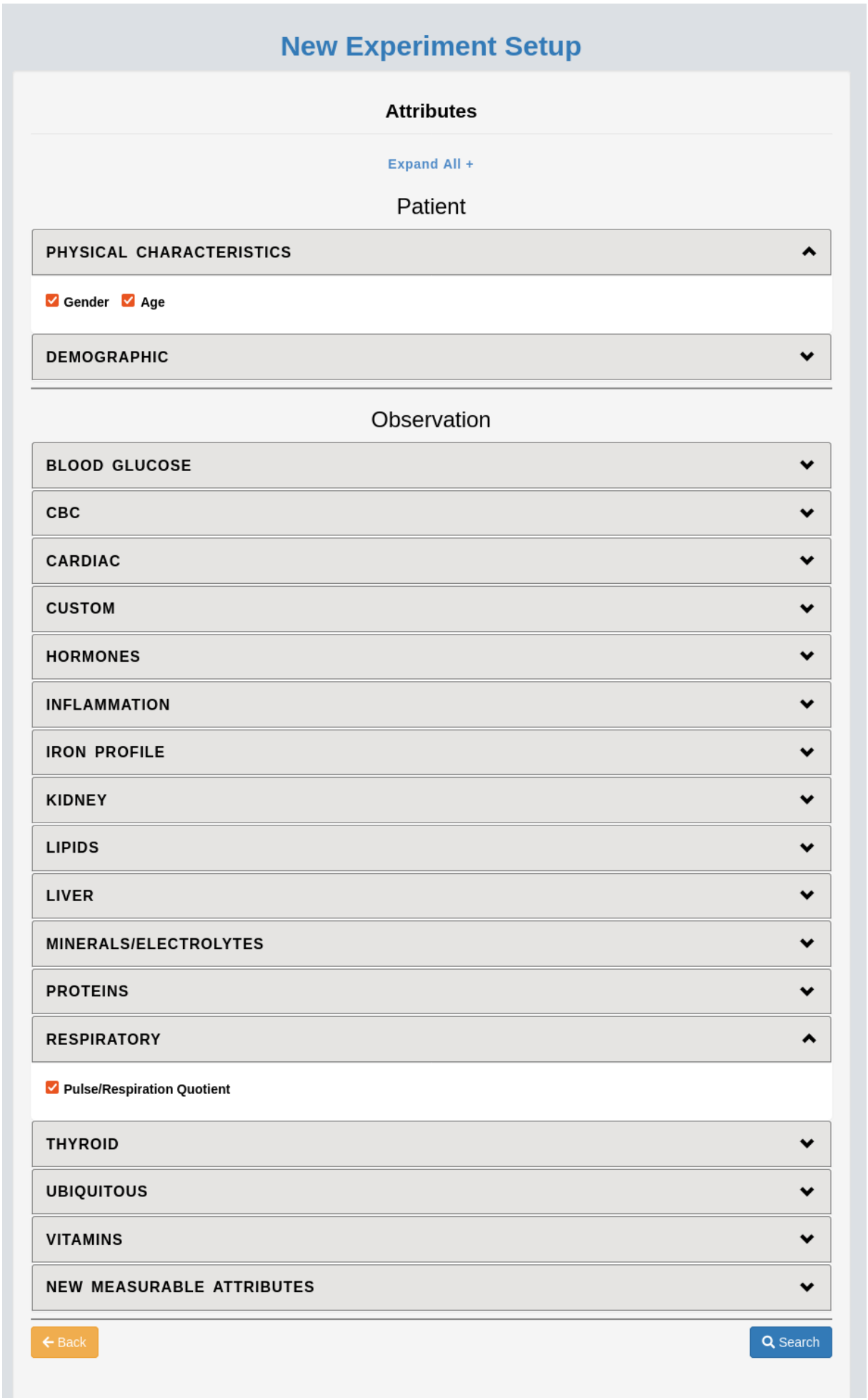
Ξεκινώντας ελέγχουμε εάν το χαρακτηριστικό app.elasticsearch είναι None και σε αυτήν την περίπτωση επιστρέφουμε την συνάρτηση χωρίς περαιτέρω δράση. Αυτό συμβαίνει ώστε όταν η υπηρεσία Elasticsearch δεν έχει ρυθμιστεί, η εφαρμογή να συνεχίζει να εκτελείται χωρίς τη δυνατότητα αναζήτησης και χωρίς να δίνει κανένα σφάλμα. Είναι απλώς θέμα ευκολίας κατά την ανάπτυξη ή κατά την εκτέλεση testing. Υπενθυμίζουμε ότι η τοποθέτηση του χαρακτηριστικού app.elasticsearch έχει επεξηγηθεί στην υποενότητα 4.2.2.3 και περιέχεται στον κώδικα 4.8. Επίσης ελέγχουμε εάν το μήκος των φίλτρων είναι μηδενικό για ευνόητους λόγους.

Η βιβλιοθήκη Elasticsearch μάς παρέχει την υλοποιημένη μέθοδο search(), την οποία θα χρησιμοποιήσουμε για την αναζήτηση. Οι παράμετροι που θα χρησιμοποιήσουμε είναι το σώμα του ερωτήματος (body), το οποίο πρέπει να χτίσουμε και το όνομα του ευρετηρίου. Η παράμετρος body πρέπει να έχει τη μορφή λεξικού. Εδώ χρειαζόμαστε τα κλειδιά query για τα φίλτρα αναζήτησης, from για τον αριθμό του στοιχείου από το οποίο θέλουμε να ξεκινήσει η ανάκτηση και size για το μέγεθος αποτελεσμάτων ανά σελίδα. Για το ερώτημα φτιάχνουμε μια νέα συνάρτηση *and_query()*, η οποία θα εφαρμόζει σύζευξη μεταξύ όλων των φίλτρων. Η βιβλιοθήκη Elasticsearch δεν παρέχει υλοποιημένο αντικείμενο σελιδοποίησης, επομένως πρέπει να κάνουμε τα μαθηματικά σελιδοποίησης για να υπολογίσουμε την τιμή from, η οποία εξαρτάται από τις αρχικές παραμέτρους page και per_page.

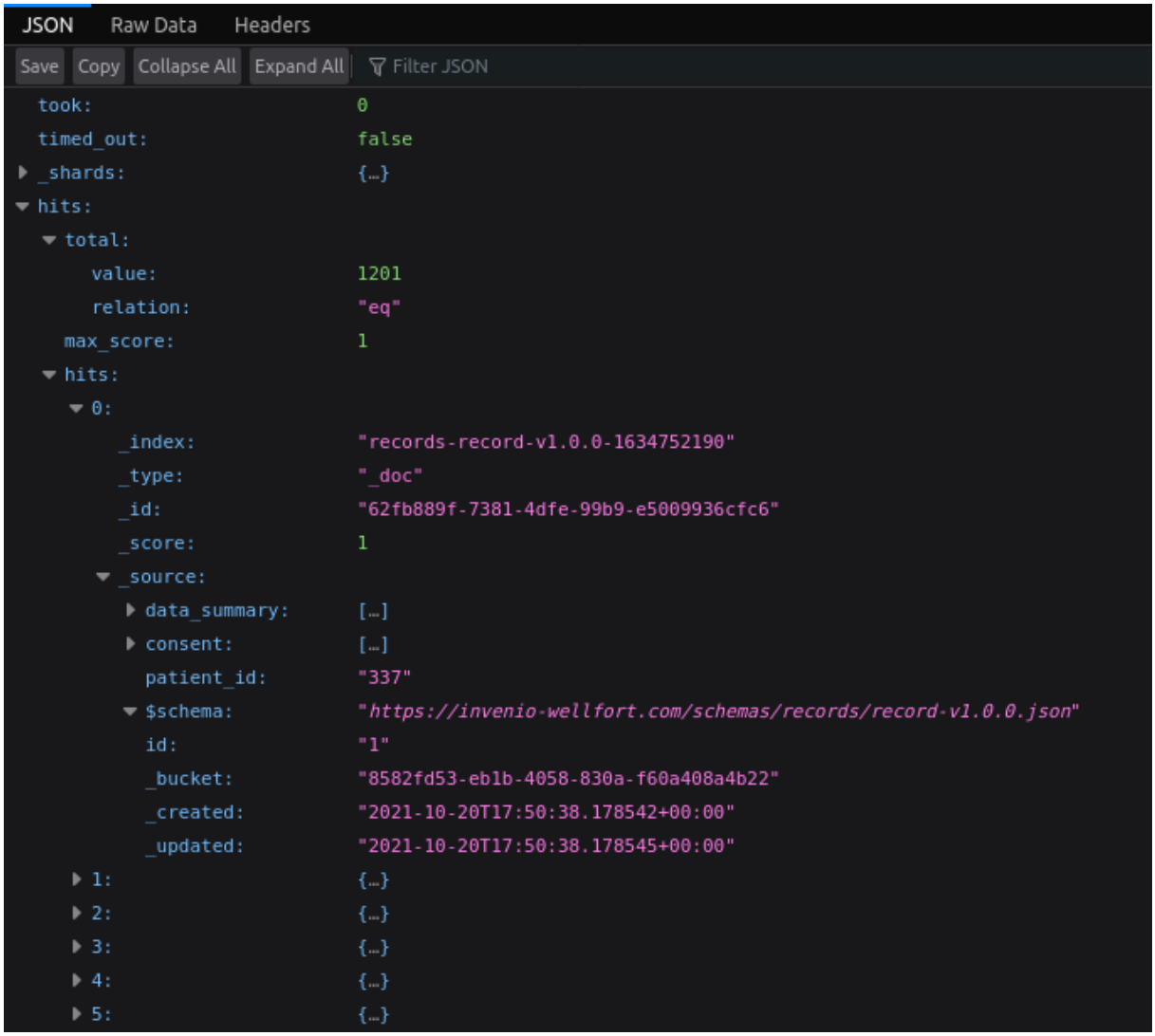
Στην εικόνα 4.6 παρουσιάζουμε ένα στιγμιότυπο αναζήτησης, ώστε να έχουμε μια γενική εικόνα της δομής που επιστρέφει η Elasticsearch. Έστω ότι το αποτέλεσμα μιας αναζήτησης παρακρατείται στην μεταβλητή search, τότε η προσπέλαση στα αποτελέσματα θα γίνεται με την έκφραση hits_obj = search[‘hits’], το οποίο είναι ένα λεξικό και περιέχει πληροφορίες σχετικά με το πλήθος των αποτελεσμάτων αλλά και τα αποτελέσματα καθεαυτά. Για να λάβουμε την λίστα των αποτελεσμάτων, αλυσιδώνουμε την έκφραση αυτή με την δήλωση hits_obj[‘hits’]. Επανερχόμενοι στην υλοποίηση, εάν η σημαία source είναι ψευδής τότε οι τιμές επιστροφής της συνάρτησης είναι δύο: η πρώτη είναι μια λίστα αναγνωριστικών για τα αποτελέσματα αναζήτησης που δημιουργούνται από την Elasticsearch, η δεύτερη είναι ο συνολικός αριθμός των αποτελεσμάτων. Όλα λαμβάνονται από το λεξικό που επιστρέφεται από τη συνάρτηση es.search(). Για τη δημιουργία των λιστών χρησιμοποιούμε *κατανόηση λίστας (list comprehension)* που μας επιτρέπει να μετατρέψουμε λίστες από τη μια μορφή στην άλλη.



Εικόνα 4.4: Απεικόνιση φόρμας για αναζήτηση δεδομένων μέσω συγκατάθεσης


New Experiment Setup
Attributes
Expand All +
Patient
PHYSICAL CHARACTERISTICS
Gender
Age
DEMOGRAPHIC
Observation
BLOOD GLUCOSE
CBC
CARDIAC
CUSTOM
HORMONES
INFLAMMATION
IRON PROFILE
KIDNEY
LIPIDS
LIVER
MINERALS/ELECTROLYTES
PROTEINS
RESPIRATORY
Pulse/Respiration Quotient
THYROID
UBIQUITOUS
VITAMINS
NEW MEASURABLE ATTRIBUTES
Back
Search

Εικόνα 4.5: Απεικόνιση φόρμας για αναζήτηση χαρακτηριστικών



Εικόνα 4.6: Στιγμιότυπο τμήματος αποτελεσμάτων της Elasticsearch

Σε αυτήν την περίπτωση, χρησιμοποιούμε την κατανόηση λίστας για να εξάγουμε τις τιμές αναγνωριστικού από την πολύ μεγαλύτερη λίστα αποτελεσμάτων που παρέχεται από την Elasticsearch. Εάν η σημαία source είναι αληθής, τότε η συνάρτηση θα επιστρέφει και τη σελιδοποιημένη λίστα με τα μεταδεδομένα ως λεξικά. Αυτό είναι χρήσιμο, όταν θέλουμε να ανακτήσουμε κάποια συγκεκριμένη ιδιότητα από τα μεταδεδομένα, όπως στην περίπτωση μας, όπου θα υπολογίζουμε τα πλήθη των εγγραφών και των ασθενών που πληρούν το δοθέν ερώτημα.

Όπως αναφέραμε, η συνάρτηση _and_query() είναι βοηθητική και εφαρμόζει το φίλτρο “KAI” μεταξύ όλων των φίλτρων, καθώς θα πρέπει να ταιριάζουμε δεδομένα που πληρούν την σύζευξη του ερωτήματος αναζήτησης. Τα φίλτρα έχουν την μορφή λεξικού τα κλειδιά των οποίων περιέχουν τα πεδία των μεταδεδομένων προς αναζήτηση, ενώ οι τιμές τους έχουν την μορφή λίστας. Με την σύνθετη κατανόηση λίστας που παρουσιάζουμε παραπάνω, οδηγούμαστε στην δημιουργία των επιθυμητών φίλτρων.

Κώδικας 4.17: *web/app/search.py* - Υλοποίηση αναζήτησης

```
from flask import current_app

def query_index(index, filters, page=1, per_page=10, source=False):
    if not current_app.elasticsearch or len(filters) == 0:
        return [], 0

    search = current_app.elasticsearch.search(
        body={
            'query': _and_query(filters),
            'from': (page - 1) * per_page,
            'size': per_page
        },
        index=index
    )
    ids = [hit['_id'] for hit in search['hits']['hits']]
    if source:
        source_list = [hit['_source'] for hit in search['hits']['hits']]
        return ids, search['hits']['total']['value'], source_list
    return ids, search['hits']['total']['value']


def _and_query(filters):
    return {
        'bool': {
            'must': [
                {'match': {path: each_v}}
                for path, values in filters.items() for each_v in values
            ]
        }
    }
```

Για την κλήση της αναζήτησης χρησιμοποιούμε έναν αόριστο βρόγχο με την συνθήκη “όσο αληθής”, την οποία θα τερματίζουμε με συγκεκριμένο έλεγχο. Η αναζήτηση επιστρέφει σελιδοποιημένα αποτελέσματα, οπότε χρειαζόμαστε τον βρόγχο για να προσπελαύνουμε κάθε σελίδα. Στις μεταβλητές ids_list και patient_count αποθηκεύουμε τις τιμές των αναγνωριστικών των εγγραφών του ΑΑΔ και των ασθενών με την κατανόηση λίστας για τα αντίστοιχα πεδία. Για να τερματίσουμε τον βρόχο ελέγχουμε εάν το πλήθος των αποτελεσμάτων είναι μικρότερο από την εκάστοτε σελίδα πολλαπλασιαζόμενη με 10, διότι αυτό είναι το προεπιλεγμένο πλήθος εγγραφών ανά σελίδα. Μόλις ο βρόχος περατωθεί, πρέπει να ελέγξουμε και το πλήθος όλων των αποτελεσμάτων. Στην περίπτωση που το πλήθος είναι μικρότερο ή ίσο του 20, τότε η επικείμενη ανάλυση θεωρείται αποκαλυπτική ως προς την ταυτότητα των ασθενών και γι’ αυτό ο αναλυτής πρέπει να υποβάλλει ένα διαφορετικό ερώτημα αναζήτησης. Τέλος υποβάλλουμε το ερώτημα αναζήτησης στη ΒΔ, αποθηκεύοντας τα φίλτρα, την λίστα αναγνωριστικών των εγγραφών και το αναγνωριστικό του χρήστη.

Κώδικας 4.18: *web/app/query/routes.py* - Ανάκτηση αποτελεσμάτων αναζήτησης

```
from app.db import db
from app.models import Query
from app.query import query_bp
from app.search import query_index

@query_bp.route('/attributes', methods=('GET', 'POST'))
@login_required
def attributes():
    # ......
    page, ids_list, patient_count = 1, list(), set()
    while True:
        _, total, meta = query_index('records', params, page,
source=True)
        ids_list.extend([source['id'] for source in meta])
        patient_count.update([source['patient_id'] for source in meta])
        if total < page * 10:
            break
        page += 1
    if total <= 20:
        return render_template('errors/400.html', message='Insufficient
amount of results')

    query = Query(data={**session['data']}, records=ids_list,
user_id=str(current_user.id))
    db.session.add(query)
    db.session.commit()
    # ......
```

4.2.2.6.2.2 Αποτελέσματα

Στην περίπτωση κατά την οποία το πλήθος των εγγραφών του ΑΑΔ που ταιριάζουν τα κριτήρια του ερωτήματος αναζήτησης είναι επαρκές, τότε ο πραγματικός αριθμός αυτών των συνόλων δεδομένων παρουσιάζεται με τον ξεχωριστό αριθμό ασθενών στους οποίους ανήκουν. Η εικόνα 4.7 απεικονίζει τη φόρμα της επισκόπησης των αποτελεσμάτων. Η πλατφόρμα προσφέρει τρεις επιλογές ασφαλούς ανάλυσης δεδομένων, με σεβασμό στο απόρρητο: ανάλυση σε συγκεντρωτική μορφή, σύνθεση και ανωνυμοποίηση δεδομένων.

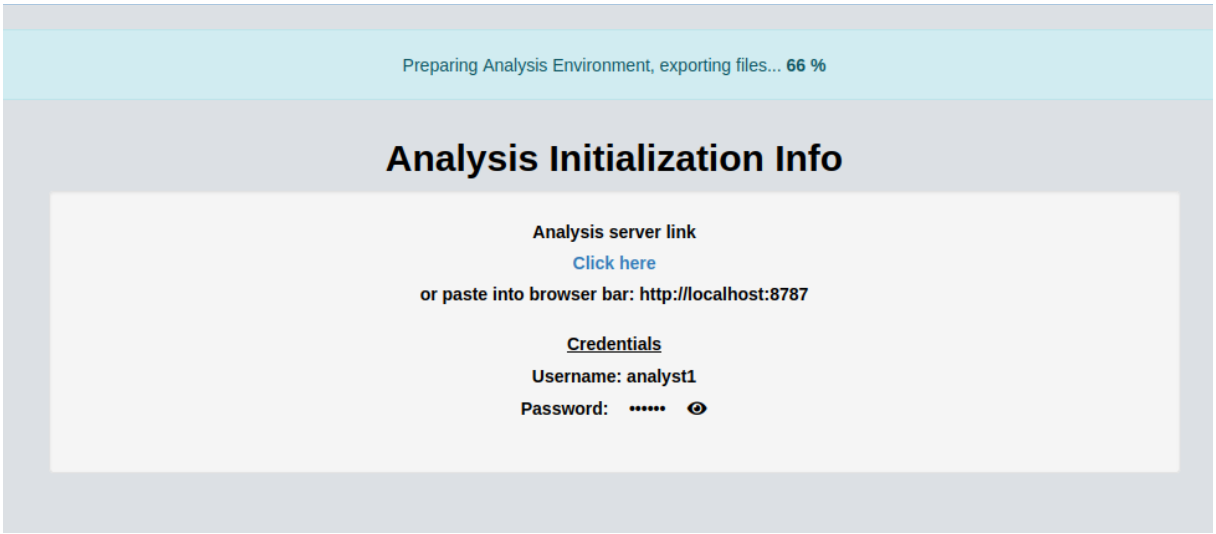
Εικόνα 4.7: Επισκόπηση αποτελεσμάτων

Η κατάσταση των κουμπιών εξαρτάται από την επιλογή επεξεργασίας που υποβλήθηκε στη φόρμα συναίνεσης. Εδώ το κουμπί ανωνυμοποίησης δεδομένων είναι φυσικά απενεργοποιημένο, καθώς προϋποθέτει την επιλογή “Anonymise” ή “Pseudo-Anonymise” στην επεξεργασία των δεδομένων. Στην αντίθετη περίπτωση που τα αποτελέσματα είναι ανεπαρκή, ο αναλυτής λαμβάνει το αντίστοιχο μήνυμα και μπορεί είτε να επιλέξει διαφορετικά χαρακτηριστικά προς εξαγωγή, είτε να ξαναξεκινήσει από την αρχή για να επιλέξει διαφορετικά φίλτρα συναίνεσης. Φυσικά, όταν τα αποτελέσματα είναι επαρκή αλλά ο αριθμός τους δεν ικανοποιεί τις ανάγκες του αναλυτή, ο τελευταίος έχει πάντα την δυνατότητα να ξεκινήσει διαφορετικό ερώτημα αναζήτησης.

4.2.2.6.3 Ανάλυση δεδομένων

Η διαδικασία για ανάλυση των δεδομένων ξεκινάει με το πάτημα του κουμπιού Initiate Analysis. Η υπηρεσία ρύθμισης πειραμάτων θα ξεκινήσει την ανάκτηση των εγγραφών από την ασφαλή αποθήκη δεδομένων βασιζόμενη στην λίστα με τα μοναδικά αναγνωριστικά των εγγραφών (persistent identifiers) που ταιριάζουν στο ερώτημα αναζήτησης που υποβλήθηκε. Κατόπιν για κάθε εγγραφή που ανακτάται, επιστρέφεται και το αρχείο στο οποίο βρίσκονται τα δεδομένα. Κατά την ανάκτηση, η υπηρεσία ξαναελέγχει ποια χαρακτηριστικά στάλθηκαν από την ΔΧ, ώστε να αποθηκεύσει τις κατάλληλες τιμές για μετατροπή. Τα αρχεία δεδομένων έχουν την μορφή JSON και μπορούν να περιέχουν αρκετές τιμές διαφόρων χαρακτηριστικών, συνεπώς πρέπει να ανακτώνται μόνο οι τιμές που ταιριάζουν στα checkboxes που μάρκαρε ο αναλυτής.

Η ανάκτηση και ο μετασχηματισμός των δεδομένων υποβάλλεται ως ασύγχρονη εργασία στην ουρά διεργασιών που περιγράψαμε στην υποενότητα 4.2.2.5. Αυτό είναι απαραίτητο, καθώς ο μετασχηματισμός μπορεί να διαρκέσει αρκετή ώρα ανάλογα με το πλήθος των δεδομένων και των χαρακτηριστικών. Το κάθε αρχείο μπορεί να έχει έως και χιλιάδες γραμμές αποθηκευμένων δεδομένων στη μορφή JSON, οι οποίες θα προσπελαύνονται σειριακά, γεγονός που θα εμποδίζει την απόκριση HTTP της εφαρμογής web έως ότου η εργασία ολοκληρωθεί. Ως εκ τούτου, η εργασία πρέπει να υποβληθεί στο παρασκήνιο, ενώ θα στέλνουμε απευθείας την απόκριση στον πελάτη.



Εικόνα 4.8: Αρχικοποίηση περιβάλλοντος ανάλυσης με ποσοστό προόδου

Ο αναλυτής πρέπει να πληροφορείται επίσης για την πρόοδο της εργασίας που υποβλήθηκε. Αυτό επιτυγχάνεται με την χρήση της ουράς μηνυμάτων, στην οποία ο worker μπορεί να εγγράψει ορισμένα μεταδεδομένα. Στην περίπτωσή μας χρησιμοποιούμε το πεδίο progress από το μοντέλο ΒΔ TaskModel

το οποίο παραθέσαμε στον κώδικα 4.15, όπου γράφουμε την ποσοστιαία πρόοδο ώστε αυτή να ανακτάται στην μεριά του πελάτη. Για την ανάκτηση της προόδου χρησιμοποιούμε μια κλήση AJAX, η οποία ζητάει την τιμή της ανά ένα ορισμένο χρονικό διάστημα.

4.2.2.6.4 Δημοσίευση δεδομένων με προστασία στο απόρρητο

Η ενότητα Δημοσίευσης Δεδομένων με προστασία στο απόρρητο υλοποιείται χρησιμοποιώντας εργαλεία ανοικτού κώδικα για την ανωνυμοποίηση δεδομένων. Υπάρχουν δύο μέθοδοι που επιτρέπουν τη λήψη δεδομένων με διατήρηση του απορρήτου: (i) δημιουργία συνθετικών δεδομένων και (ii) k-ανωνυμοποίηση (*k-anonymization*) [6].

4.2.2.6.4.1 Σύνθεση δεδομένων

Τα συνθετικά σύνολα δεδομένων χρησιμοποιούνται για τη συμπλήρωση, αύξηση και σε ορισμένες περιπτώσεις όπως του συστήματός μας, την αντικατάσταση πραγματικών δεδομένων κατά την εκπαίδευση μοντέλων Μηχανικής Μάθησης. Έτσι, εξαλείφεται ο κίνδυνος έκθεσης των δεδομένων, ο οποίος συνοδεύει την αποκάλυψη της ταυτότητας.

Τα συνθετικά σύνολα δεδομένων δημιουργούνται χρησιμοποιώντας το Synthetic Data Vault (SDV) στο στοιχείο *Δημοσίευσης Δεδομένων με Προστασία στο Απόρρητο* (βλ. σχήμα 3.2). Το SDV είναι μια βιβλιοθήκη Python που βασίζεται σε πιθανολογική γραφική μοντελοποίηση (*Probabilistic Graphical Modeling*) και βαθιά μηχανική μάθηση (*Deep Learning*) για τη δημιουργία συνθετικών δεδομένων που έχουν την ίδια μορφή και τις ίδιες στατιστικές ιδιότητες με τα αρχικά δεδομένα.

Στον κώδικα 4.20 παρουσιάζουμε ένα κομμάτι της δημιουργίας συνθετικών συνόλων δεδομένων χρησιμοποιώντας την βιβλιοθήκη SDV. Αρχικά εισάγουμε τις απαραίτητες βιβλιοθήκες όπως την pandas για την ανάγνωση αρχείων csv και τις κλάσεις SDV και Metadata από την βιβλιοθήκη SDV. Η λογική του κώδικα είναι η εξής: Για κάθε αρχείο csv (π.χ. ένα αρχείο με τους ασθενείς και ένα αρχείο με τις μετρήσεις) διαβάζουμε τα δεδομένα ως dataframes, από τη συνάρτηση read_csv() της βιβλιοθήκης pandas. Έπειτα ελέγχουμε εάν ο τρέχων πίνακας είναι τύπου ασθενή (αλλιώς θα είναι τύπου παρατήρησης-observation καθώς δουλεύουμε μόνο με αυτές τις 2 περιπτώσεις). Στην περίπτωση πίνακα ασθενών περνάμε το dataframe σε ένα νέο λεξικό και γράφουμε τα μεταδεδομένα του sdv όπου περνάμε το όνομα του πίνακα, τις τιμές και το αναγνωριστικό ως κύριο κλειδί (το οποίο είναι πάντα σταθερό ως στήλη στον πίνακα ασθενών).

Κώδικας 4.19: *synthesize.py* - Σύνθεση δεδομένων

```
import os
import pandas as pd
from sdv import SDV, Metadata

tables_to_synth = dict()
metadata = Metadata()
sdv = SDV()

for table_name in data_files:
    input_df = pd.read_csv(self.data_files[table_name],
```

```
skipinitialspace=True)

    if table_name == 'patient':
        tables_to_synth[table_name] = input_df
        metadata.add_table(name=table_name, data=input_df,
primary_key='id')
    else:
        dfs_map = _split_dfs(input_df, table_name)
        for name in dfs_map:
            tables_to_synth[name] = dfs_map[name]
            metadata.add_table(
                name=name,
                data=dfs_map[name],
                parent='patient',
                foreign_key='patient_id'
            )
    metadata.to_json(os.path.join(study_dir, 'meta.json'))
    sdv.fit(metadata, tables_to_synth)
    samples = sdv.sample()
    obs_to_merge = list()
    for table in tables_to_synth:
        if 'observation' in table:
            obs_to_merge.append(samples[table])
            continue
        final_df = samples[table]
        final_df.to_csv(os.path.join(self.project_dir, 'synthetic_data_'
+ table + '.csv'), index=False)
    pd.concat(obs_to_merge,
ignore_index=True).to_csv(os.path.join(self.project_dir,
'synthetic_data_observation.csv'), index=False)
```

Στην περίπτωση πίνακα παρατηρήσεων, πρέπει να διαχωρίσουμε τα dataframes καθώς μερικές στήλες πρέπει να μην υποστούν σύνθεση (όπως το ξένο κλειδί του ασθενή που είναι παρόν στις παρατηρήσεις για την αντιστοίχιση). Η συνάρτηση _split_dfs() παραλείπεται για λόγους συντομίας. Με τον ίδιο τρόπο περνάμε στο λεξικό προς σύνθεση μόνο τις στήλες που πρέπει να την υποστούν με κλειδί το όνομα του πίνακα. Για λόγους ελέγχου γράφουμε τα μεταδεδομένα σε μορφή JSON σε ένα αρχείο, το οποίο θα περιέχει πληροφορίες π.χ. σχετικά με τους πίνακες, τις στήλες τους και τους τύπους των τιμών τους. Για να πραγματοποιηθεί η σύνθεση πρέπει πρώτα να καλέσουμε την μέθοδο fit() του sdv αντικειμένου, η οποία ουσιαστικά εκπαιδεύει το μοντέλο μας. Ο ακριβής χρόνος που χρειάζεται η εκπαίδευση αυτή εξαρτάται από πολλούς παράγοντες, συμπεριλαμβανομένου του αριθμού των σειρών, των στηλών και των διακριτών κατηγοριών στις κατηγορικές στήλες. Μόλις αποκτήσουμε το μοντέλο μας, μπορούμε να αρχίσουμε να δημιουργούμε συνθετικά δεδομένα χρησιμοποιώντας τη μέθοδο sample(). Τυπικά η μέθοδος αυτή λαμβάνει ως παράμετρο τον αριθμό γραμμών που θέλουμε να έχουμε στο αποτέλεσμα. Παραλείποντάς την, υπονοούμε ότι τα αποτελέσματα θα έχουν ίδιο πλήθος γραμμών με τα αρχικά

σύνολα δεδομένων. Ο υπόλοιπος κώδικας ελέγχει εάν υπάρχει πίνακας παρατηρήσεων στα δεδομένα προς σύνθεση και ξαναπροσθέτει τις γραμμές που διαχωρίστηκαν στο τελικό αποτέλεσμα.

	A	B	C	D	E
1	value	biomarker	biomarker_code	issued	patient_id
2	5.58	Total Cholesterol	TCH	2010-12-13T03:14:04Z	409
3	5.45	Total Cholesterol	TCH	1982-09-13T06:37:06Z	452
4	3.46	Total Cholesterol	TCH	1991-10-26T23:14:01Z	240
5	5.01	Total Cholesterol	TCH	1985-04-01T05:21:25Z	377
6	5.28	Total Cholesterol	TCH	1980-03-22T18:26:48Z	293
7	4.59	Total Cholesterol	TCH	1999-07-25T22:33:37Z	232
8	4.87	Total Cholesterol	TCH	1984-04-22T16:42:01Z	461
9	5.83	Total Cholesterol	TCH	1982-12-25T08:39:12Z	280
10	5.35	Total Cholesterol	TCH	1994-03-20T17:53:12Z	150
11	4.26	Total Cholesterol	TCH	2008-01-02T16:17:22Z	384
12	2.89	Total Cholesterol	TCH	1980-05-08T08:53:40Z	209
13	4.91	Total Cholesterol	TCH	2008-10-10T14:18:09Z	4
14	6.23	Total Cholesterol	TCH	2019-10-23T07:25:05Z	492

Εικόνα 4.9: Τμήμα αρχικού συνόλου δεδομένων παρατηρήσεων

Η εικόνα 4.9 παρουσιάζει μέρος των συνόλων δεδομένων πριν τη σύνθεση, ενώ η εικόνα 4.10 το ίδιο τμήμα του συνόλου δεδομένων μετά τη σύνθεση. Παίρνοντας ως σημείο αναφοράς την τιμή id σε κάθε γραμμή, παρατηρούμε ότι το μοντέλο μας μεταβάλλει επιτυχώς τις τιμές και τα στιγμιότυπα των παρατηρήσεων, οπότε πληρείται η διατήρηση της ιδιωτικότητας.

	A	B	C	D	E
1	value	issued	biomarker	biomarker_code	patient_id
2	4.87	1985-04-01T05:21:25Z	Total Cholesterol	TCH	409
3	5.45	1982-09-13T06:37:06Z	Total Cholesterol	TCH	452
4	5.28	1985-04-01T05:21:25Z	Total Cholesterol	TCH	240
5	4.26	2008-01-02T16:17:22Z	Total Cholesterol	TCH	377
6	2.89	1989-11-24T14:24:12Z	Total Cholesterol	TCH	293
7	2.89	1980-05-08T08:53:40Z	Total Cholesterol	TCH	232
8	4.75	1985-04-01T05:21:25Z	Total Cholesterol	TCH	461
9	4.87	1980-03-22T18:26:48Z	Total Cholesterol	TCH	280
10	4.91	1980-03-22T18:26:48Z	Total Cholesterol	TCH	150
11	4.75	1984-04-22T16:42:01Z	Total Cholesterol	TCH	384
12	7.24	2019-10-23T07:25:05Z	Total Cholesterol	TCH	209
13	8.1	2005-11-09T06:23:05Z	Total Cholesterol	TCH	4
14	5.45	1982-09-13T06:37:06Z	Total Cholesterol	TCH	492

Εικόνα 4.10: Τμήμα συνόλου δεδομένων παρατηρήσεων μετά την σύνθεση

4.2.2.6.4.2 Ανωνυμοποίηση δεδομένων

Για την k-ανωνυμοποίηση χρησιμοποιούμε το ARX Data Anonymization Tool. Το εργαλείο αυτό επιτρέπει την επέκταση της εγγύησης απορρήτου των ληφθέντων δεδομένων υποστηρίζοντας πολλαπλά

πρόσθετα μοντέλα απορρήτου δεδομένων, όπως l-diversity, t-closeness κ.λπ. Η επιλογή των αναγνωριστικών και quasi-αναγνωριστικών γίνεται μέσα στην πλατφόρμα και βασίζεται σε λεξιλόγια με κατηγορίες. Τα άμεσα αναγνωριστικά περιγράφονται από την κατηγορία Identifying και αποκρύπτονται για τη λήψη δεδομένων. Τα quasi-χαρακτηριστικά είναι εκείνα που εμπίπτουν σε ένα υποσύνολο της κατηγορίας *Προσωπικά Δεδομένα* όπως *Εθνικότητα*, *Γεωγραφικά Στοιχεία* (π.χ. διεύθυνση κατοικίας) και *Φυσικά Χαρακτηριστικά*.

	A	B	C
30	29	20s	<30
31	30	30s	[30-50)
32	31	30s	[30-50)
33	32	30s	[30-50)
34	33	30s	[30-50)
35	34	30s	[30-50)
36	35	30s	[30-50)
37	36	30s	[30-50)
38	37	30s	[30-50)
39	38	30s	[30-50)
40	39	30s	[30-50)
41	40	40s	[30-50)
42	41	40s	[30-50)
43	42	40s	[30-50)
44	43	40s	[30-50)
45	44	40s	[30-50)
46	45	40s	[30-50)
47	46	40s	[30-50)
48	47	40s	[30-50)
49	48	40s	[30-50)
50	49	40s	[30-50)
51	50	50s	[50-70)
52	51	50s	[50-70)
53	52	50s	[50-70)
54	53	50s	[50-70)

Εικόνα 4.11: Παράδειγμα κανόνων γενικοποίησης για την ηλικία

Η υλοποίηση της ανωνυμοποίησης γίνεται με την Python βιβλιοθήκη *crowds* και με τον ορισμό ιεραρχιών προς γενικοποίηση. Ένα τέτοιο παράδειγμα παρατίθεται στον εικόνα 4.11, όπου έχουμε ένα αρχείο csv με διάφορα εύρη ηλικιών. Στην πρώτη στήλη τοποθετούμε την τιμή της ηλικίας και κατόπιν ορίζουμε το πως θα γενικευτεί. Η δεύτερη στήλη περιέχει κανόνα γενίκευσης χρησιμοποιώντας δεκαετίες. Για παράδειγμα αν κάποιος ασθενής είναι 35 χρονών, τότε η γενίκευση θα είναι 30s. Αυτός ο κανόνας θεωρείται το 1ο επίπεδο γενίκευσης. Τέλος, η τρίτη στήλη περιέχει μεγαλύτερα διαστήματα ηλικιών, όπου εστιάζουμε με διαστήματα στις ηλικίες 30 έως 70, ενώ σε ηλικίες μικρότερες του 30 και μεγαλύτερες του 70, παραθέτουμε απλώς <30 και >70 αντίστοιχα. Η υλοποίηση του κώδικα για την ανωνυμοποίηση είναι πολύπλοκη και για αυτό το λόγο δεν θα την αναλύσουμε περαιτέρω.

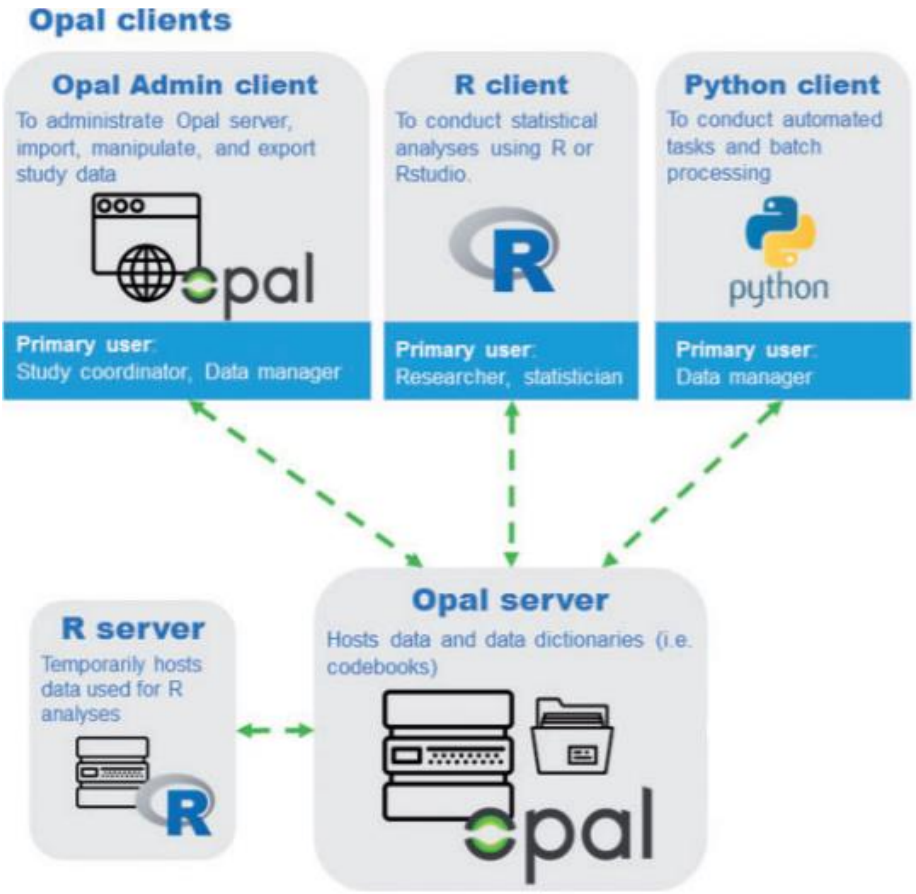
4.3 Υλοποίηση ασφαλούς περιβάλλοντος ανάλυσης δεδομένων

Το ΑΠΑΔ περιέχει τους μηχανισμούς για την ασφαλή ανάλυση δεδομένων και τη διατήρηση της ιδιωτικότητας των χρηστών, στους οποίους ανήκουν τα δεδομένα. Ο μηχανισμός αυτός απαρτίζεται από

τον διακομιστή ανάλυσης, τη ΒΔ ανάλυσης, όπου αποθηκεύονται προσωρινά τα δεδομένα της μελέτης και τη διεπαφή ανάλυσης, η οποία επιτρέπει στον αναλυτή να αλληλεπιδρά με τα δεδομένα.

4.3.1 Υλοποίηση διακομιστή ανάλυσης δεδομένων

Για αυτό το στοιχείο, βασιζόμαστε στην τεχνολογία Opal [47], το οποίο είναι ένα σύστημα διαχείρισης δεδομένων που παρέχει εργαλεία για εισαγωγή, μετατροπή, περιγραφή και εξαγωγή δεδομένων. Οι διαχειριστές δεδομένων μπορούν να εισάγουν με ασφάλεια μια ποικιλία τύπων δεδομένων, π.χ. κείμενο, αριθμητικά δεδομένα, εικόνες κ.λπ. και μορφές αρχείων, π.χ., SPSS ή CSV. Το Opal είναι χτισμένο σε υπηρεσίες ιστού REST, οι οποίες επιτρέπουν συνδέσεις από διαφορετικούς πελάτες web λογισμικού χρησιμοποιώντας το πρωτόκολλο HTTPS, όπως φαίνεται στην εικόνα 4.12.



Εικόνα 4.12: Σύνδεση του διακομιστή Opal με τους υπόλοιπους πελάτες

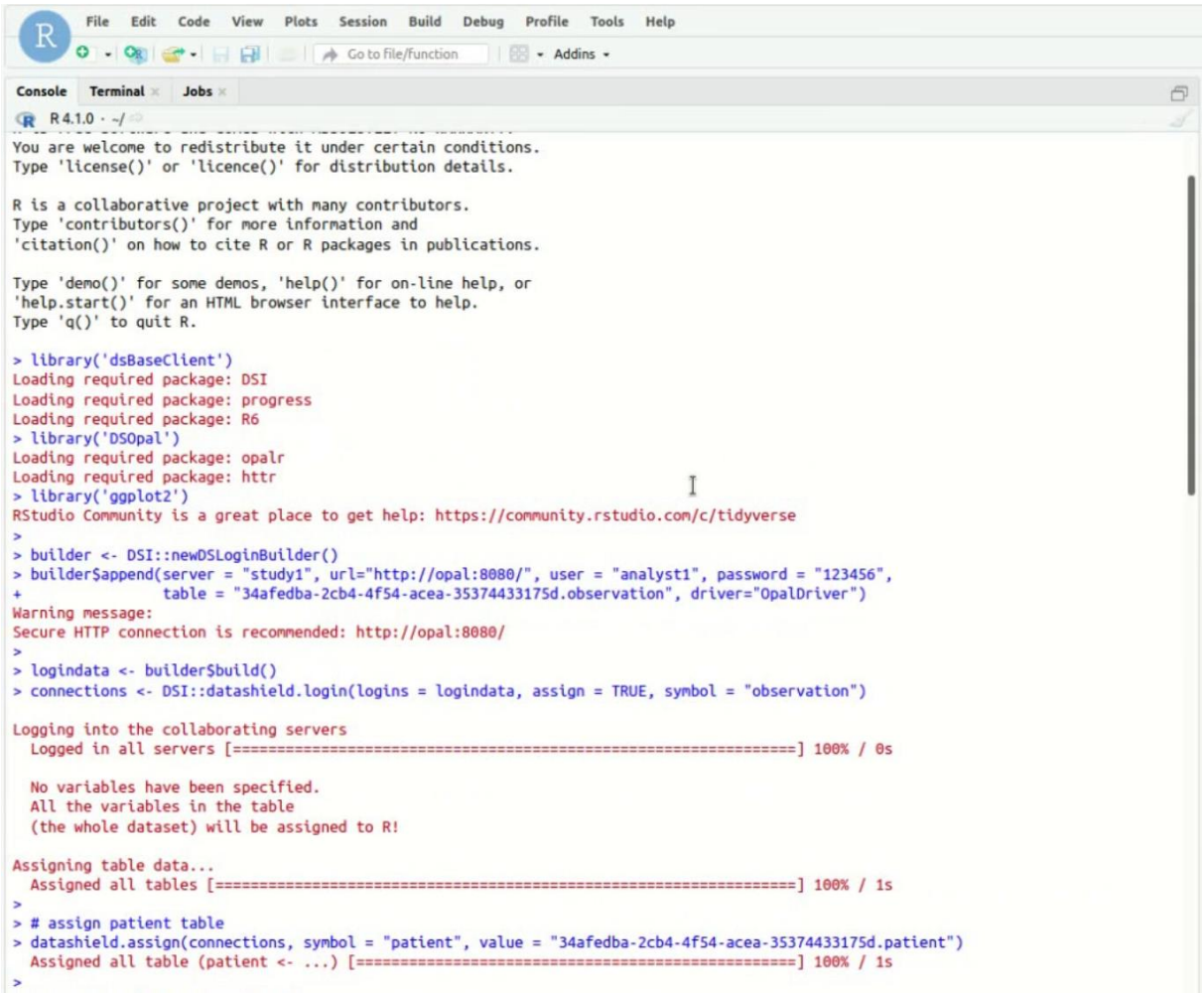
Αυτοί οι πελάτες έχουν σχεδιαστεί για να διασφαλίζουν ένα ευρύ φάσμα εξειδικευμένων εργασιών που μπορούν να επιτευχθούν από διαφορετικούς χρήστες. Για παράδειγμα, ένας πελάτης Python επιτρέπει στους διαχειριστές δεδομένων (σε αυτήν την περίπτωση στον διαχειριστή της πλατφόρμας μας) να αυτοματοποιούν επαναλαμβανόμενες εργασίες όπως η εισαγωγή δεδομένων και η διαχείριση αδειών πρόσβασης. Ένας πελάτης R, όπως το RStudio, επιτρέπει τη διεξαγωγή στατιστικής ανάλυσης των δεδομένων που είναι αποθηκευμένα στο Opal. Στην περίπτωση μας, ο αναλυτής θα μπορεί να προσπελάσει τα μετασχηματισμένα δεδομένα μόνο μέσω της διεπαφής του RStudio, η οποία αναλύεται στην υποενότητα 4.2.3.2

Το Opal παρέχει πρόσβαση στη ΒΔ ανάλυσης δεδομένων στους επαληθευμένους χρήστες. Ο *Φορτωτής* ορίζει τα δικαιώματα χρήστη για τους πίνακες δεδομένων που έχουν φορτωθεί στη *Βάση Δεδομένων Ανάλυσης Δεδομένων*, έτσι ώστε ο Αναλυτής να μπορεί να έχει πρόσβαση σε αυτούς με τη χρήση της *Διεπαφής Ανάλυσης Δεδομένων*. Αυτό επιτρέπει στους αναλυτές να εργάζονται μόνο με τα δεδομένα που έχουν ανακτηθεί για μελέτες που ξεκίνησαν.

Ο έλεγχος ταυτότητας πελάτη πραγματοποιείται χρησιμοποιώντας όνομα χρήστη/κωδικό πρόσβασης.

4.3.2 Υλοποίηση διεπαφής ανάλυσης δεδομένων

Στο πρωτότυπό μας, χρησιμοποιούμε τον πελάτη R του Opal ως διεπαφή όπου ο αναλυτής διεξάγει τις μελέτες της σε επιλεγμένα δεδομένα χρησιμοποιώντας πακέτα R DataSHIELD. Το DataSHIELD [4] είναι ένα λογισμικό ανοιχτού κώδικα που παρέχει ένα τροποποιημένο στατιστικό περιβάλλον R που συνδέεται με μια ΒΔ του διακομιστή ανάλυσης Opal.



Εικόνα 4.13: Διεπαφή ανάλυσης δεδομένων RStudio

Αναπτύχθηκε αρχικά για απομακρυσμένη, μη αποκαλυπτική ανάλυση δεδομένων βιοϊατρικής, υγειονομικής περίθαλψης και κοινωνικών επιστημών, αν και μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον όπου τα μικροδεδομένα (δεδομένα σε ατομικό επίπεδο) πρέπει να αναλυθούν, αλλά δεν μπορούν ή δεν πρέπει να κοινοποιηθούν φυσικά με τους χρήστες για διάφορους ηθικούς και νομικούς λόγους ή όταν τα αντικείμενα δεδομένων είναι πολύ μεγάλα για να τα μοιραστούν.

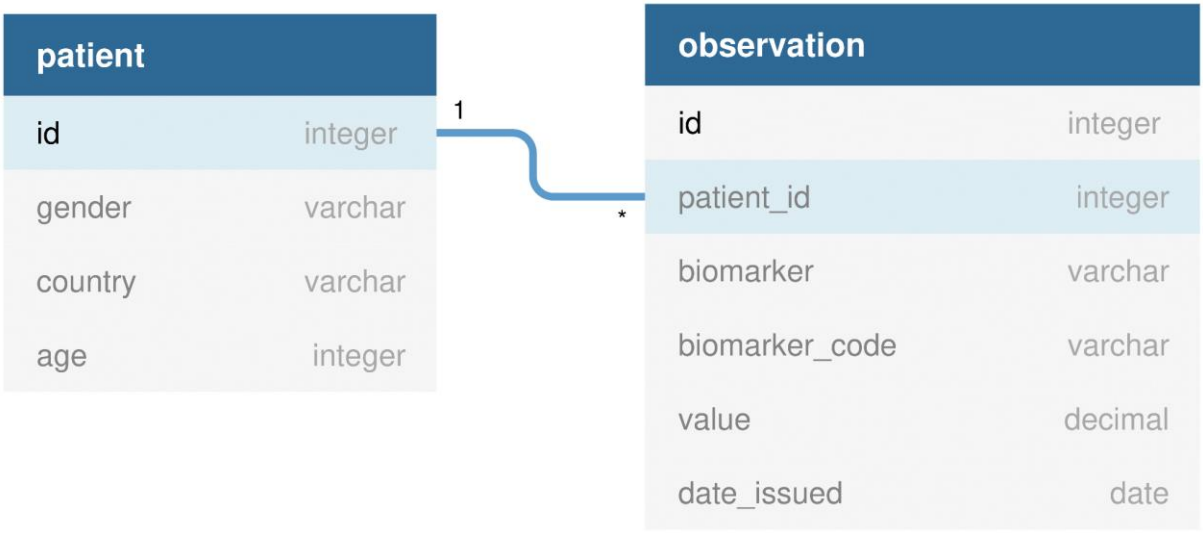
Το DataSHIELD εφαρμόζει περισσότερες από εκατό στατιστικές μεθόδους που ενσωματώνουν παγίδες προστασίας της ιδιωτικότητας, οι οποίες εμποδίζουν τον αναλυτή να εντοπίσει μεμονωμένα υποκείμενα δεδομένων ή να συναγάγει την τιμή συγκεκριμένων μεταβλητών στα δεδομένα με βάση αναλυτικά αποτελέσματα. Για παράδειγμα, το subsetting περιορίζεται λειτουργικά καθώς δεν δημιουργείται κανένα υποσύνολο δεδομένων που περιέχει από προεπιλογή, μόνο 1-4 παρατηρήσεις. Αυτό συμβαίνει γιατί τα αποτελέσματα που βασίζονται σε αυτά τα υποσύνολα ενδέχεται να αποκαλύπτουν πληροφορίες.

Ο διακομιστής RStudio χρησιμοποιείται ως διεπαφή χρήστη της πλατφόρμας μας, την οποία ένας αναλυτής χρησιμοποιεί για να συνδεθεί στο διακομιστή Opal και να πραγματοποιήσει ανάλυση διατήρησης απορρήτου μέσω λειτουργιών DataSHIELD. Η διεπαφή του RStudio φαίνεται στην εικόνα 4.13, όπου εμπεριέχονται και κομμάτια κώδικα R για την διεξαγωγή ενός πειράματος.

4.3.3 Υλοποίηση ΒΔ ανάλυσης δεδομένων

Στη ΒΔ ανάλυσης δεδομένων χρησιμοποιούμε το Fast Healthcare Interoperability Resources (FHIR), που χρησιμοποιείται από το Health Level 7 International (HL7), ως το πρότυπο της μορφής των δεδομένων. Φυσικά, αυτή η μορφή χρησιμοποιείται και στο αποθετήριο δεδομένων. Το HL7 FHIR είναι το πιο πρόσφατο πρότυπο της HL7 για την ηλεκτρονική ανταλλαγή πληροφοριών υγειονομικής περίθαλψης με βάση προσεγγίσεις αναδυόμενων βιομηχανιών [48]. Το FHIR προσφέρει μια σειρά από πλεονεκτήματα για την ανάπτυξη πρωτοτύπων, όπως η σταθερή βάση του στα πρότυπα Web και η υποστήριξη για αρχιτεκτονικές RESTful.

Στην περίπτωσή μας, χρησιμοποιούμε το FHIR για να αντιπροσωπεύσουμε τον *Ασθενή* (*Patient*) και την *Παρατήρηση* (*Observation*). Ενώ το FHIR συνιστά τη χρήση του LOINC [49] για δεδομένα παρατήρησης, χρησιμοποιούμε προσαρμοσμένους κωδικούς ταξινόμησης για πρακτικούς λόγους.



Εικόνα 4.14: Απεικόνιση σχήματος Βάσης Δεδομένων για ανάλυση δεδομένων

Το ΑΠΑΔ διατηρεί την ιδιωτικότητα, γεγονός που συνεπάγεται σε μείωση της χρησιμότητας των δεδομένων και της ευελιξίας μιας ανάλυσης σε σύγκριση με την πλήρη πρόσβαση σε δεδομένα. Μια άλλη πρόκληση είναι ότι τα δεδομένα που προέρχονται από διαφορετικές πηγές πρέπει να μετατραπούν στην ίδια κοινή μορφή, διατηρώντας παράλληλα τα δεδομένα ομοιόμορφα για να εργαστούν οι αναλυτές. Επομένως, στη ΒΔ ανάλυσης δεδομένων στοχεύουμε να διατηρήσουμε τα πιο σχετικά δεδομένα για την ανάλυση σε μια μάλλον συνοπτική αλλά περιεκτική μορφή. Η βάση δεδομένων αποτελείται από δύο προεπιλεγμένους πίνακες που ορίζονται από τους τύπους πόρων FHIR: *Patient* (*Ασθενής*) και *Observation* (*Παρατήρηση*). Ο πίνακας Patient παρέχει πληροφορίες σχετικά με τους χρήστες που περιλαμβάνονται στη μελέτη, όπως φύλο, ηλικία και χώρα. Ο πίνακας Observation περιέχει πληροφορίες σχετικά με τις συγκεκριμένες μετρήσεις για έναν χρήστη. Ένας βιοδείκτης μπορεί να είναι οποιοδήποτε χαρακτηριστικό που περιέχεται στις βάσεις δεδομένων πηγής, για παράδειγμα, σάκχαρο στο αίμα, επίπεδο δραστηριότητας κ.λπ. Κάθε βιοδείκτης αντιπροσωπεύεται με τον κωδικό του

(*biomarkerCode*) και το πλήρες όνομά του (*biomarker*). Ένας ασθενής μπορεί να έχει πολλαπλές μετρήσεις του ίδιου βιοδείκτη που μετρώνται σε διαφορετική χρονική στιγμή, επομένως το χαρακτηριστικό *issued* παρέχει τις πληροφορίες για το πότε έχει ληφθεί η συγκεκριμένη μέτρηση. Οι δύο πίνακες συνδέονται με το αναγνωριστικό του χρήστη: Observation:patientId - Patient:id. Το σχήμα της βάσης δεδομένων φαίνεται στην εικόνα 4.14. Όπως ορίζεται από το Opal, κάθε πίνακας αντιπροσωπεύεται από (i) ένα αρχείο CSV που ονομάζεται πίνακας δεδομένων (*data table*) και (ii) ένα αρχείο JSON που ονομάζεται λεξικό δεδομένων (*data dictionary*). Η τεχνολογία που επιλέξαμε για τον διακομιστή της ΒΔ ανάλυσης δεδομένων είναι η Postgresql.

4.4 Επίλογος

Στο παρόν κεφάλαιο παραθέσαμε την υλοποίηση του συστήματος μας, κομμάτια της οποίας απαρτίζουν το πρωτότυπο μας. Χωρίσαμε την υλοποίηση σε δύο βασικά μέρη, σύμφωνα με το Σχήμα 3.2: Την υλοποίηση της *Ασφαλούς Αποθήκης Δεδομένων* και την υλοποίηση του *Ασφαλούς Περιβάλλοντος Ανάλυσης Δεδομένων*. Μαζί με κάθε υλοποίηση, παρουσιάσαμε τις τεχνολογίες που χρησιμοποίησαμε και αποσπάσματα κώδικα με επεξηγήσεις. Το κεφάλαιο που ακολουθεί παρουσιάζει την αξιολόγηση του συστήματος με ΠΧ και μετρήσεις χρόνου εκτέλεσης κώδικα.

Κεφάλαιο 5ο: Αξιολόγηση

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο, παρουσιάζουμε την αξιολόγηση της πλατφόρμας μέσω αντιπροσωπευτικών περιπτώσεων χρήσης. Επιπρόσθετα αξιολογούμε τις επιδόσεις του πηγαίου κώδικα στους μετασχηματισμούς των δεδομένων. Στη σύνοψη αξιολογούμε πόσο καλά απαντούν τα σενάρια που επιδεικνύονται στις απαιτήσεις που τίθενται στο κεφάλαιο 3.

5.2 Δεδομένα

Για την αξιολόγηση χρησιμοποιούμε δεδομένα που προέρχονται από δύο πηγές, αμφότερες από ΜΜΕ (μικρομεσαίες εταιρείες) που παρέχουν στους τελικούς χρήστες μια εφαρμογή σχετική με την υγειονομική περίθαλψη ή τον τρόπο ζωής. Για λόγους εμπιστευτικότητας, ονομάζουμε αυτές τις πηγές ως εταιρία (ή Πηγή δεδομένων) Α και εταιρία Β. Στον Πίνακα 5.1 παρουσιάζεται ένα υποσύνολο των χαρακτηριστικών που παρέχονται από αυτές τις δύο πηγές δεδομένων.

Πίνακας 5.1: Χαρακτηριστικά των δεδομένων αξιολόγησης

Πηγή δεδομένων Α		Πηγή δεδομένων Β	
Χαρακτηριστικό	Τύπος	Χαρακτηριστικό	Τύπος
Φύλο	συμβολοσειρά	Φύλο	συμβολοσειρά
Εθνικότητα	συμβολοσειρά	Ημερομηνία Γέννησης	ημερομηνία
Ηλικία	φυσικός αριθμός	Καρδιακός ρυθμός	δεκαδικός αριθμός
Λιπίδια	δεκαδικός αριθμός	Δείκτης ζωτικότητας	δεκαδικός αριθμός
Νεφρό	δεκαδικός αριθμός	Πηλίκο παλμού αναπνοής	δεκαδικός αριθμός
Συκώτι	δεκαδικός αριθμός	Συμπαθητική δραστηριότητα	δεκαδικός αριθμός
Γλυκόζη αίματος	δεκαδικός αριθμός	Παρασυμπαθητική δραστηριότητα	δεκαδικός αριθμός

Δημιουργήσαμε 1100 συνθετικά σύνολα δεδομένων για 600 χρήστες με βάση τα αρχικά δεδομένα και από τις δύο εταιρείες με διαφορετικές διαμορφώσεις συναίνεσης. Κάθε σύνολο δεδομένων περιέχει μια ενιαία εγγραφή για δεδομένα ασθενούς (π.χ. ταυτότητα, ηλικία και χώρα) και χαρακτηριστικά παρατήρησης υγείας (βλ. Πίνακα 5.1) που παρατηρούμε στις περιπτώσεις χρήσης για μία ημέρα.

5.3 Περιπτώσεις Χρήσης

Ορίζουμε τρεις περιπτώσεις χρήσης που θα διεκπεραιωθούν στην πλατφόρμα:

- ΠΧ1: Περιγραφική ανάλυση δεδομένων από μία μόνο πηγή δεδομένων

- ΠΧ2: Περιγραφική ανάλυση δεδομένων από πολλαπλές πηγές δεδομένων
- ΠΧ3: Δημοσίευση δεδομένων

Οι ΠΧ 1 και 2 περιγράφουν σενάρια χρήσης του ασφαλούς περιβάλλοντος ανάλυσης, ενώ η ΠΧ3 περιγράφει τον σκοπό του στοιχείου ΔΔΠΑ. Η ΠΧ1 ασχολείται με δεδομένα που προέρχονται από μία μόνο πηγή δεδομένων, ενώ παρουσιάζει το φάσμα των λειτουργιών που παρέχονται στον αναλυτή. Η ΠΧ2 εστιάζει σε μια μελέτη που χρησιμοποιεί δεδομένα από πολλαπλές πηγές.

Σε κάθε ΠΧ, παρέχουμε επίσης μια συζήτηση σχετικά με ζητήματα απορρήτου των αντίστοιχων αναλύσεων.

Προσομοιώνουμε το ακόλουθο σενάριο για να εισαγάγουμε δυναμικές πτυχές:

- ➢ Αρχική κατάσταση: όλα τα σύνολα δεδομένων χρήστη μεταφορτώνονται και συλλέγονται στην Ασφαλή αποθήκη δεδομένων.
- ➢ Ημέρα 1 (01-03-2022): Εκτελούνται οι ΠΧ 1 και 2. Η ρύθμιση ανάλυσης δεδομένων για την ΠΧ1 περιέχει τα ακόλουθα φίλτρα: λήξη συναίνεσης: 01-03-2022 και επεξεργασία: Ανάλυση, Διανομή. Αποδίδει 450 σύνολα δεδομένων συνολικά από την εταιρία Β. Για την ρύθμιση ανάλυσης δεδομένων της ΠΧ2 ορίζονται τα εξής φίλτρα: λήξη συναίνεσης: 03-03-2022, επεξεργασία: Ανάλυση, Διανομή, Συνδυασμός. Αποδίδει 800 σύνολα δεδομένων συνολικά, 400 το καθένα από την εταιρία Α και την εταιρία Β.
- ➢ Ημέρα 2 (02-03-2022): Η συναίνεση χρήστη από 50 χρήστες έληξε (χρήστης 0 έως 49) λόγω της ημερομηνίας. Επιπλέον, 50 χρήστες (χρήστης 100 έως 149) αφαίρεσαν την επεξεργασία Διανομή από τη συγκατάθεσή τους.
- ➢ Ημέρα 3 (03-03-2022): Η ΠΧ 3 εκτελείται. Λόγω των αλλαγών και της λήξης ορισμένων συναινέσεων των χρηστών, η ρύθμιση της ανάλυσης για την ΠΧ3 με τα φίλτρα: λήξη συναίνεσης: 03-03-2022 και επεξεργασία: Ανάλυση, Διανομή αποδίδει μόνο 350 σύνολα δεδομένων από την εταιρία Β, η οποία είναι μικρότερη από την ΠΧ1 με πανομοιότυπη ρύθμιση αξιολόγησης (στην επεξεργασία).

Η ροή για κάθε ΠΧ μοιράζεται ορισμένα κοινά βήματα:

- Βήμα 1: Ο **αναλυτής** ξεκινά τη μελέτη του επιλέγοντας σχετικά χαρακτηριστικά, προσδιορίζοντας τον σκοπό και την ημερομηνία λήξης της μελέτης.
- Βήμα 2: Η **πλατφόρμα** αποδίδει τα διαθέσιμα σύνολα δεδομένων σύμφωνα με:
    - τη συναίνεση χρήστη που πρέπει να ταιριάζει με το σκοπό μελέτης του Αναλυτή
    - την ημερομηνία λήξης της συγκατάθεσης των δεδομένων που πρέπει να είναι μεταγενέστερη από την ημερομηνία λήξης της μελέτης.

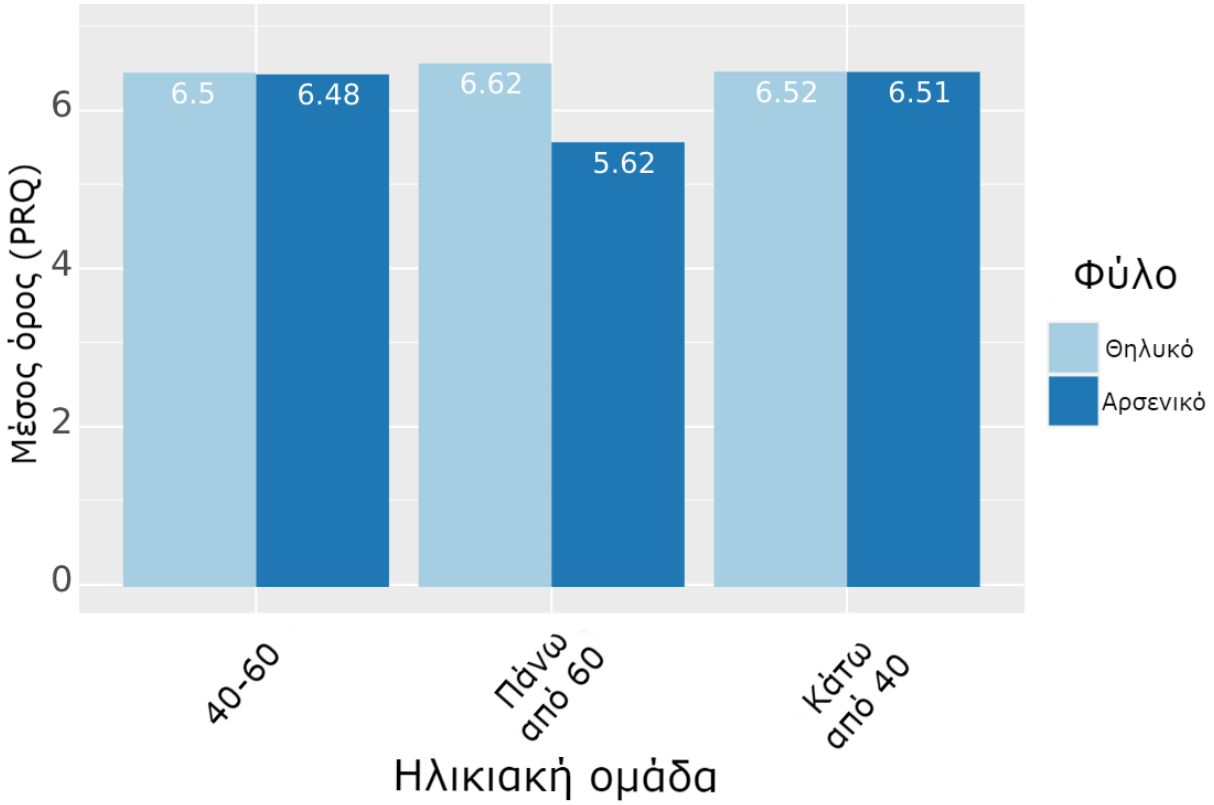
Ο αριθμός των διαθέσιμων συνόλων δεδομένων πρέπει να υπερβαίνει το *προκαθορισμένο κατώφλι*, διαφορετικά η ανάλυση θεωρείται δυνητικά αποκαλυπτική προς την ταυτότητα των χρηστών οπότε και η πλατφόρμα τερματίζει τη ρύθμιση της μελέτης. Για τους σκοπούς της αξιολόγησης, ορίσαμε το *ελάχιστο κατώφλι* στην τιμή 10.

- Βήμα 3: Για κάθε μελέτη κατόπιν της αρχικοποίησης, η **πλατφόρμα** μετατρέπει τα διαθέσιμα σύνολα δεδομένων σε αρχεία που περιγράφουν δύο σχεσιακούς πίνακες, ονομάτων *Ασθενής* και *Παρατήρηση*, όπως επεξηγήθηκε στην υποενότητα 4.3.3.
- Βήμα 4: Για κάθε μελέτη, η **πλατφόρμα** ορίζει τα δικαιώματα χρήστη στον διακομιστή Opal για τον αναλυτή που επιτρέπει τη χρήση δεδομένων μέσω του πακέτου DataSHIELD. Με αυτόν τον τρόπο κάθε αναλυτής βλέπει μόνο τις μελέτες που ξεκίνησε.

- Βήμα 5: Όταν ξεκινά η μελέτη, ο **αναλυτής** χρησιμοποιεί:
    - είτε το ΑΠΑΔ στο οποίο συνδέεται με τα διαπιστευτήρια του. Εκεί γίνεται χρήση του πακέτου DataSHIELD χρησιμοποιώντας ακόμα μία φορά τα διαπιστευτήριά του, ώστε να συνδεθεί στο διακομιστή Opal μέσω αυτού και να εκτελέσει την ανάλυση.
    - είτε το ΔΔΠΑ για να κάνει λήψη των μετασχηματισμένων δεδομένων του.

5.3.1 ΠΧ1: Περιγραφική ανάλυση δεδομένων από μία πηγή

Στην πρώτη ΠΧ, υποθέτουμε ότι ο αναλυτής θέλει να πραγματοποιήσει ένα σύνολο περιγραφικών μελετών για να απεικονίσει επιλεγμένες τιμές χαρακτηριστικών σε όλο τον πληθυσμό των χρηστών της πλατφόρμας. Έστω αυτό το χαρακτηριστικό Pulse/Respiration-Quotient (PRQ) που προέρχεται από τη ΒΔ της εταιρίας Β. Στη συνέχεια, ο αναλυτής θέλει να μελετήσει πώς διαφέρουν οι τιμές PRQ μεταξύ ηλικιακών ομάδων και φύλου. Στην αξιολόγησή μας, η πλατφόρμα βρίσκει 450 συμμετέχοντες με ένα σύνολο εγγραφών των επιλεγμένων χαρακτηριστικών, επομένως ο πίνακας Patient περιέχει 450 εγγραφές ηλικίας και φύλου των χρηστών και ο πίνακας Observation 450 εγγραφές του PRQ.



Σχήμα 5.1: ΠΧ1 - Συσχέτιση ηλικιακών ομάδων με το μέσο όρο PRQ

Το σχήμα 5.1 δείχνει τα αποτελέσματα της ανάλυσης, δηλαδή τον μέσο όρο του PRQ για διαφορετικές ηλικιακές ομάδες (κάτω των 40, 40-60 και άνω των 60 ετών) για άνδρες και γυναίκες ξεχωριστά. Οι μέσοι όροι του PRQ λαμβάνονται από τις συναρτήσεις DataSHIELD και στη συνέχεια σχεδιάζονται χρησιμοποιώντας το πακέτο σχεδίασης R *ggplot*. Ο αναλυτής χρησιμοποιεί ένα σύνολο χειρισμών δεδομένων και στατιστικών συναρτήσεων από το πακέτο DataSHIELD, όπως για παράδειγμα:

- Σύνδεση πινάκων μέσω της συνάρτησης cbind: ο αναλυτής πρέπει να ενώσει τους πίνακες Patient και Observation μέσω του αναγνωριστικού.

- Δημιουργία υποσυνόλου μέσω της συνάρτησης dataFrameSubset: αυτή η συνάρτηση χρησιμοποιείται για να πραγματοποιήσει υποσύνολα των ενωμένων πινάκων από το προηγούμενο βήμα με τις τιμές της ηλικίας και φύλου..
- Συνάρτηση μέσης τιμής μέσω της συνάρτησης mean: ο αναλυτής χρησιμοποιεί αυτή τη συνάρτηση για να υπολογίσει τη μέση τιμή κάθε υποσυνόλου.

Σε αυτό το σημείο παραθέτουμε μία μικρή συζήτηση για την ιδιωτικότητα. Οι συναρτήσεις DataSHIELD που αναφέρονται για την δημιουργία υποσυνόλων και τον υπολογισμό της μέσης τιμής είναι δυνητικά αποκαλυπτικές ως προς την ταυτότητα των χρηστών συναρτήσεις. Για παράδειγμα, η επανειλημμένη εφαρμογή υποσυνόλων μπορεί να οδηγήσει στην αποκάλυψη των πραγματικών τιμών από τον πίνακα. Το DataSHIELD επιτυγχάνει εκδόσεις αυτών των λειτουργιών που διατηρούν το απόρρητο, απαγορεύοντας τα υποσύνολα και τον υπολογισμό απλών στατιστικών, όπως ο μέσος όρος, σε λιγότερα από 4 δείγματα δεδομένων. Η λειτουργικότητα για τη σύνδεση πινάκων υλοποιείται έτσι ώστε η μέθοδος να μην επιστρέφει καμία τιμή στον αναλυτή, εκτός από πιθανά μηνύματα σφάλματος. Ο ενωμένος πίνακας παραμένει στον διακομιστή σε μια μεταβλητή, την οποία ο αναλυτής μπορεί να χρησιμοποιήσει σε άλλες συναρτήσεις μόνο ονομαστικά, επομένως δεν μπορούν να αποκαλυφθούν τιμές στη διαδικασία.

5.3.2 ΠΧ2: Περιγραφική ανάλυση δεδομένων από πολλαπλές πηγές

Στην ΠΧ2 ο αναλυτής θέλει να πραγματοποιήσει μια μελέτη σχετικά με τα χαρακτηριστικά που προέρχονται από ξεχωριστές πηγές δεδομένων. Για παράδειγμα, ο αναλυτής θέλει να συγκρίνει τους καρδιακούς παλμούς ηρεμίας ατόμων με διαφορετικά προφίλ δίαιτας, όπως η χορτοφαγική διατροφή, η δίαιτα χαμηλή σε υδατάνθρακες κ.λπ. Για αυτό το σκοπό, η πλατφόρμα πρέπει να συλλέξει τα δεδομένα από δύο εφαρμογές. Η εταιρία Α παρέχει το χαρακτηριστικό *καρδιακός ρυθμός* και η εταιρεία Β παρέχει το χαρακτηριστικό *δίαιτα*. Αυτή η ανάλυση είναι δυνατή μόνο σε χρήστες που χρησιμοποιούν και τις δύο εφαρμογές. Οι χρήστες ελέγχουν τη σύνδεση των δεδομένων τους και πρέπει να ενεργοποιήσουν τη σύνδεση μεταξύ αυτών των εφαρμογών κατά την υποβολή δεδομένων (στις εφαρμογές τους).

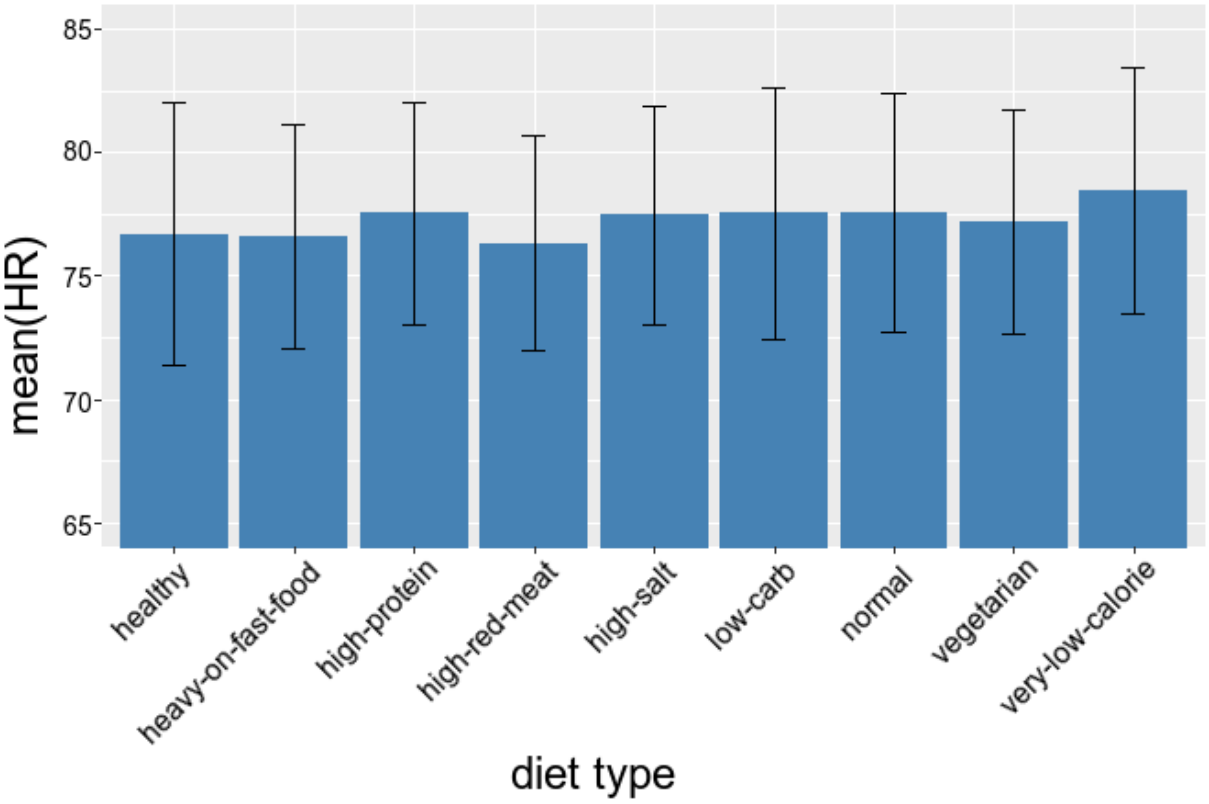
Στην τρέχουσα αξιολόγηση, ο πίνακας *Patient* περιέχει 400 εγγραφές χρηστών και ο πίνακας *Observation* περιέχει 800 εγγραφές για αυτούς τους 400 χρήστες. 400 εγγραφές καρδιακών παλμών και 400 τύπου δίαιτας. Ο Αναλυτής ομαδοποιεί τους ασθενείς με βάση τον τύπο διατροφής τους και, στη συνέχεια, για κάθε ομάδα δίαιτας υπολογίζει τη μέση και τυπική απόκλιση των τιμών του καρδιακού ρυθμού.

Για να επισημάνουμε μερικά βήματα, απαριθμούμε ορισμένες συναρτήσεις DataSHIELD που επιτρέπουν αυτές τις λειτουργίες:

- Υποσύνθεση μέσω της συνάρτησης *ds.dataFrameSubset*: διαχωρισμός του πίνακα Observation ανάλογα με το χαρακτηριστικό "biomarker" έτσι ώστε κάθε υποπίνακας να περιέχει έναν τύπο βιοδείκτη, δηλαδή έναν πίνακα για τη διατροφή, έναν άλλο πίνακα για τον καρδιακό ρυθμό.
- Σύνδεση πινάκων μέσω της συνάρτησης *ds.merge*: ένωση των υποπινάκων από το προηγούμενο βήμα στο χαρακτηριστικό *patient_id* και με αυτόν τον τρόπο περιστροφή του πίνακα *Observation* σε έναν πίνακα με το νέο σύνολο στηλών: [patient_id, diet_value, HR_value, ...]
- Συνάθροιση (aggregation) μέσω της συνάρτησης *ds.meanSdGp*: δεδομένου ότι ο αναλυτής υποσυνθέτει τον περιστρεφόμενο πίνακα από το προηγούμενο βήμα με το χαρακτηριστικό

diet_value, αυτή η συνάρτηση υπολογίζει τη μέση και τυπική απόκλιση των τιμών του καρδιακού ρυθμού για κάθε ομάδα δίαιτας.

Το αποτέλεσμα της παραπάνω ανάλυσης που προκύπτει εμφανίζεται στο σχήμα 5.2. Σε αυτήν την εικόνα απεικονίζεται η μέση και τυπική απόκλιση της τιμής του καρδιακού ρυθμού για άτομα σε διαφορετικές ομάδες δίαιτας. Από την άποψη της διατήρησης του απορρήτου, η συνάρτηση *ds.merge* επιτυγχάνεται με τον ίδιο τρόπο που αναφέρθηκε προηγουμένως της συνάρτησης *ds.cbind*, χωρίς να επιστρέφει κανένα αποτέλεσμα στον αναλυτή. Οι υλοποιήσεις των συναρτήσεων συνάθροισης επιτυγχάνονται με παρόμοιο τρόπο όπως οι υποσυνθέσεις, με την απαγόρευση λειτουργιών σε λιγότερα από 4 δείγματα δεδομένων.



Σχήμα 5.2: ΠΧ2 - Μέση και τυπική απόκλιση καρδιακού ρυθμού ατόμων διαφορετικής δίαιτας

5.3.3 ΠΧ3: Δημοσίευση δεδομένων

Σε αυτήν την περίπτωση χρήσης, ο αναλυτής θέλει να κατεβάσει ανωνυμοποιημένα δεδομένα, ώστε να μπορεί να πραγματοποιήσει ανάλυση εκτός σύνδεσης σε αυτά, χρησιμοποιώντας άλλα εργαλεία ανάλυσης δεδομένων αντί του ΑΠΑΔ. Οι λόγοι για αυτό μπορεί να είναι τεχνικής φύσης, όπως προτιμήσεις γλώσσας προγραμματισμού, εξοικείωση και εξάρτηση από άλλο framework ανάλυσης, απαιτούμενες λειτουργίες που δεν καλύπτονται από το DataSHIELD κ.λπ. Επιπλέον, ο αναλυτής μπορεί να ενδιαφέρεται για τη δομή και για τις συγκεκριμένες τιμές αυτών των δεδομένων αντί της στατιστικής ανάλυσης ή να επιθυμεί να αποφύγει τον χρονικό περιορισμό για τη διεξαγωγή των μελετών. Ο αναλυτής, ωστόσο, πρέπει να έχει επίγνωση για την απώλεια ποιότητας και συνεπώς χρησιμότητας που εκδηλώνουν τα ανωνυμοποιημένα δεδομένα.

Ο αναλυτής επιλέγει τα χαρακτηριστικά *φύλο, ηλικία, χώρα* και *ολική χοληστερόλη* που παρέχονται από την Εταιρεία Μ. Στο ανωνυμοποιημένο σύνολο δεδομένων, τα άμεσα αναγνωριστικά, όπως το πλήρες όνομα χρήστη ή οποιοδήποτε είδος αναγνωριστικών, είναι φυσικά απόντα. Άλλα χαρακτηριστικά που

περιέχουν προσωπικά στοιχεία, γενικεύονται σύμφωνα με τις ιεραρχίες που έχουν προκαθοριστεί από την πλατφόρμα. Για παράδειγμα τα χαρακτηριστικά χώρα, ηλικία και φύλο χρήστη, τα οποία μπορούν να χρησιμοποιηθούν έμμεσα για ταυτοποίηση, αποκαλούμενα ως ψευδο-αναγωριστικά, ακολουθούν μια συγκεκριμένη ιεραρχία. Ένα παράδειγμα μιας τέτοιας ιεραρχίας είναι το ακόλουθο:

- *φύλο*
  - Επίπεδο 1: * (πλήρως γενικευμένο)
- *ηλικία*
  - Επίπεδο 1: [20,40), [40,60), [60, 81)
  - Επίπεδο 2: [20, 81)
- *χώρα*
  - Επίπεδο 1: ΝΕ (Νότια Ευρώπη), ΝΑΕ (Νοτιοανατολική Ευρώπη), ΚΕ (Κεντρική Ευρώπη), κ.λ.π.
  - Επίπεδο 2: * (πλήρως γενικευμένο)

Επιπλέον, οι ακριβείς τιμές των χαρακτηριστικών που δεν μπορούν να χρησιμοποιηθούν για ταυτοποίηση χρηστών, όπως η *ολική χοληστερόλη*, παραμένουν στην αρχική τους μορφή. Η δυνατότητα επαναπροσδιορισμού ατόμων από τη ΒΔ με βάση αυτές τις πληροφορίες είναι ελάχιστη. Η εικόνα 5.1 εμφανίζει ένα υποσύνολο δεδομένων μετά την εφαρμογή της k-ανωνυμοποίησης αντίστοιχα, όπου το τελευταίο είναι το αποτέλεσμα αυτής της ανάλυσης.

	A	B	C	D	E	F
1	index	id	age	gender	language	country
2	0	337	60s		en	East Europe
3	21	561	60s		en	East Europe
4	47	536	60s		en	East Europe
5	66	386	60s		en	East Europe
6	115	492	60s		en	East Europe
7	139	436	60s		en	East Europe
8	149	203	60s		en	East Europe
9	158	447	60s		en	East Europe
10	163	57	60s		en	East Europe
11	207	287	60s		en	East Europe

Εικόνα 5.1: k-ανωνυμοποίηση με k=10

5.4 Αξιολόγηση απαιτήσεων

Για να συνοψίσουμε το πρώτο κομμάτι της αξιολόγησης, αξιολογούμε την πλατφόρμα σύμφωνα με τις απαιτήσεις που ορίζονται στο κεφάλαιο 2, στοχεύοντας συγκεκριμένα στοιχεία της πλατφόρμας.

Στον πίνακα 5.2, παραθέτουμε ρητά τις απαιτήσεις και υποδεικνύουμε σε ποιο βαθμό ικανοποιούνται από το περιγραφόμενο σενάριο: πλήρως, μερικώς ή μη ικανοποιημένες. Οι απαιτήσεις 1, 2, 3 και 4 ικανοποιούνται πλήρως από την πλατφόρμα, κάτι που φαίνεται από τις ΠΧ τις οποίες περιγράψαμε παραπάνω. Τα δεδομένα για ανάλυση αποθηκεύονται στη μετασχηματισμένη μορφή μόνο στη ΒΔ ανάλυσης και τα αποτελέσματα της ανάλυσης εμφανίζονται στον αναλυτή και δεν αποθηκεύονται καθόλου στην πλατφόρμα. Και οι δύο αυτές μορφές δεδομένων αποτελούν μέρος του ΑΠΑΔ και

διαγράφονται μαζί με το δημιουργηθέν περιβάλλον, το οποίο ορίζεται από την ημερομηνία λήξης της μελέτης. Η ΠΧ1 υποστηρίζει άμεσα την απαίτηση 4. Ωστόσο, η απαίτηση 5 αντιμετωπίζει περιορισμούς όσον αφορά τη σύνδεση των χρηστών σε διαφορετικές πλατφόρμες, όπως έχει συζητηθεί στην ΠΧ2. Η πλατφόρμα μας προσφέρει την υποδομή για συνδυασμό δεδομένων με διαφορετικά σχήματα δεδομένων που προέρχονται από διαφορετικές πηγές. Παρ'όλα αυτά, οι μέθοδοι σύνδεσης εγγραφών κρύβουν ορισμένες ανησυχίες σχετικά με το απόρρητο και την ακρίβεια, επομένως δίνουμε στον χρήστη τον πλήρη έλεγχο της σύνδεσης των δεδομένων του. Καθώς η λύση μας εξαρτάται από την συμμετοχή των χρηστών, την αξιολογούμε ως μερικώς ικανοποιημένη.

Πίνακας 5.2: Αξιολόγηση των απαιτήσεων

Α/Α	**Απαίτηση**	**πλήρης**	**μερική**	**ανικανοποίητη**
1	Τα δεδομένα εισόδου και τα ενδιάμεσα δεδομένα δεν διατηρούνται στην πλατφόρμα.	●		
2	Όλοι οι υπολογισμοί στα αρχικά δεδομένα λαμβάνουν χώρα μέσα στην πλατφόρμα	●		
3	Η πλατφόρμα περιορίζει τη χρονική περίοδο για την αποθήκευση του περιβάλλοντος ανάλυσης	●		
4	Ανάλυση από μία μόνο πηγή δεδομένων.	●		
5	Ανάλυση από πολλαπλές πηγές δεδομένων.		●	
6	Δεν υπάρχει λήψη δεδομένων στην αρχική μορφή.	●		
7	Λήψη ανωνυμοποιημένων δεδομένων	●		
8	Λήψη συνθετικών δεδομένων	●		
9	Μέτρα για την απαγόρευση λήψης παρόμοιων δεδομένων			●
10	Έλεγχος πρόσβασης χρήστη	●		
11	Η προέλευση των δεδομένων ανιχνεύεται σε ολόκληρο τον κύκλο ζωής των δεδομένων	●		
12	Η συγκατάθεση των χρηστών για μελέτες εξακολουθεί να ισχύει μετά την ανάκληση της συγκατάθεσής τους	●		

13	Για προηγούμενες μελέτες διατηρούνται ελάχιστες μετα-πληροφορίες	●		

Επιπλέον, οι απαιτήσεις 6, 7 και 8 περιγράφονται μέσω της ΠΧ3 και η υλοποίησή τους παραθέτεται λεπτομερώς στην υποενότητα 4.2.2.6.4. Η απαίτηση 9 παραμένει ανικανοποίητη καθώς δεν έχουν υλοποιηθεί μέτρα απαγόρευσης δημοσίευσης παρόμοιων δεδομένων. Η απαίτηση αυτή έχει ως στόχο να αντιμετωπίσει επιθέσεις απορρήτου, όταν ένας κακόβουλος χρήστης έχει πρόσβαση σε πολλαπλά σύνολα δεδομένων, είτε παρομοίως ανωνυμοποιημένα είτε συνθετικά. Θα παρατεθεί ως μέρος του επόμενου κεφαλαίου.

Οι υπόλοιπες απαιτήσεις αφορούν την προέλευση των δράσεων που λαμβάνουν χώρα στο σύστημα. Η απαίτηση 11 ικανοποιείται πλήρως στην πλατφόρμα, όπως φαίνεται μέσω της αυτοματοποιημένης καταγραφής των ιχνών προέλευσης και των δυνατοτήτων διαχείρισης και ανάλυσης προέλευσης με τα αρχεία καταγραφής, όπως αναλύθηκε στο κεφάλαιο 4. Η ΑΑΔ και το ΑΠΑΔ στέλνουν αρχεία καταγραφής σε προκαθορισμένη μορφή, η οποία στη συνέχεια αντιστοιχίζεται στο μοντέλο προέλευσης. Οι απαιτήσεις 12 και 13 αφορούν προηγούμενες μελέτες. Μόνο δεδομένα με έγκυρη συγκατάθεση χρηστών αναλύονται εντός της πλατφόρμας. Ωστόσο ενδέχεται οι χρήστες, των οποίων τα δεδομένα έχουν αναλυθεί, να ανακαλέσουν τη συγκατάθεσή τους μετά τη διεξαγωγή μιας μελέτης. Σε αυτήν την περίπτωση, τα ισχύοντα δεδομένα θα αφαιρεθούν και θα παραμείνουν μόνο δείκτες αναφοράς για προηγούμενες μελέτες.

5.5 Βελτίωση Κώδικα

Έπειτα από πολλαπλές δοκιμές, παρατηρήσαμε ότι ο κώδικας της Υπηρεσίας Ρυθμίσεων Πειραμάτων παρουσιάζει κάποια προβλήματα επιδόσεων. Πιο συγκεκριμένα, η εφαρμογή απαιτεί μεγάλους χρόνους στην ανάκτηση και επεξεργασία των αρχείων δεδομένων. Υπενθυμίζουμε, σύμφωνα και με την ενότητα 3, πως η ανάκτηση των δεδομένων πραγματοποιείται από την ΥΡΠ μέσω του ΑΑΔ, η οποία είναι και ο μόνος τρόπος για να προσπελαστούν τα δεδομένα σε συγκεντρωτική μορφή. Για κάθε μοναδικό αναγνωριστικό εγγραφής, η ΥΡΠ λαμβάνει το αντίστοιχο αρχείο, το οποίο και ανοίγει ως προς ανάγνωση.

Κώδικας 5.1: Απόσπασμα σειριακής ανάκτησης και μετατροπής δεδομένων

```
def export_csvs(records, data, data_files, dir):
    i = 0
    # 1. loop through de-serialized list
    for record in records:
        # 2. for the record ID fetch file ID through Invenio API
        record_resp =
get(url='{}/api/records/{}/files'.format(app.config['SECURE_REPO_URL'],
record), verify=False)
        record_json = record_resp.json()

        if record_resp.status_code == 200:
            # Under the assumption that 1 file is attached to the record!
```

```
            for field in record_json['contents']:
                file_id = field.get('file_id')
                filename = field['key']
                # 3. download file
                resp = get(
                    url='{}/api/records/{}/files/{}'.format(
                        app.config['SECURE_REPO_URL'],
                        record,
                        filename
                    ),
                    stream=True,
                    verify=False
                )
            json_data = json.loads(resp.text)
            # Loop through entry object
            for entry in json_data.get('entry'):
                # Initialize row with index
                row_dict = dict()
                # processing and export
                # ...
                append_dict_as_csv_row(
                    file_name=os.path.join(
                        EXPORT_FOLDER,
                        self.project_dir,
                        self.data_files[resource_type] + '.tmp'
                    ),
                    row_dict=row_dict
                )
                i += 1
                _set_task_progress(100 * i // len(records))
```

Ο κώδικας 5.1 αποτελεί απόσπασμα της ροής αυτής. Καθώς τα αρχεία μπορούν να περιέχουν πληροφορίες για χαρακτηριστικά τα οποία δεν έχουν επιλεγεί από τον αναλυτή, η σειριακή αναζήτηση των περιεχομένων τους μπορούν να οδηγήσουν σε μεγάλους χρόνους εξαγωγής αποτελεσμάτων. Αυτό είναι ιδιαίτερα παρατηρήσιμο όταν το πλήθος των αρχείων προς ανάγνωση και εξαγωγή είναι μεγάλο (>1000).

Παραπάνω παραθέτουμε την συνάρτηση export_csvs() η οποία είναι υπεύθυνη για την μετατροπή των δεδομένων που πληρούν ένα ερώτημα αναζήτησης του αναλυτή σε αρχείο(α) csv. Ο κώδικας ακολουθεί την εξής λογική: Για κάθε εγγραφή ανακτούμε το αντίστοιχο αρχείο, το ανοίγουμε για ανάγνωση, ψάχνουμε τα αντίστοιχα στοιχεία που αναζητήθηκαν και τα προσθέτουμε ως νέα γραμμή στο αρχείο csv.

Μια απλή και γρήγορη λύση είναι να χρησιμοποιήσουμε πολυεπεξεργασία στο βρόγχο για κάθε αναγνωριστικό εγγραφής. Η *πολυεπεξεργασία* αναφέρεται σε ένα σύστημα που έχει περισσότερες από δύο κεντρικές μονάδες επεξεργασίας (CPU). Κάθε επιπλέον ΚΜΕ που προστίθεται σε ένα σύστημα αυξάνει την ταχύτητα, την ισχύ και τη μνήμη του. Αυτό μας επιτρέπει να εκτελούμε πολλές διεργασίες

ταυτόχρονα. Τονίζουμε πως η λύση αυτή δεν είναι γενικευμένη, καθώς εξαρτάται από τον αριθμό των ΚΜΕ που θα διαθέτει ο εξυπηρετητής στον οποίο εκτελείται ο κώδικας.

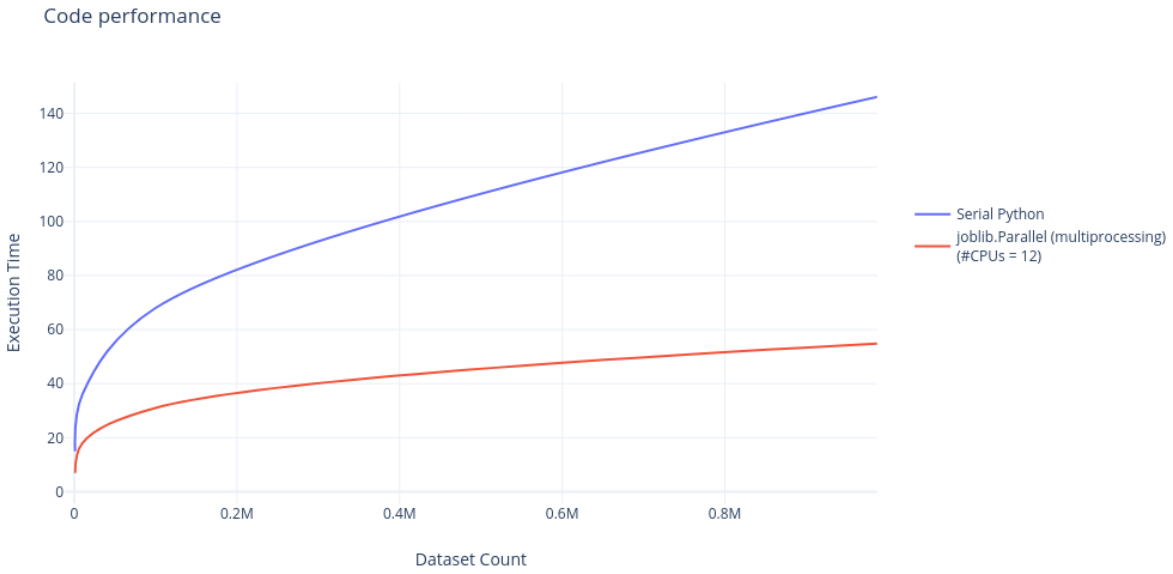
Στον κώδικα 5.2 παρουσιάζουμε την έκδοση του κώδικα που χρησιμοποιούμε για ανάκτηση και μετατροπή των δεδομένων με πολυεπεξεργασία. Σε γενικές γραμμές, η πολυεπεξεργασία παρέχει απλή κωδικοποίηση που είναι εύκολα κατανοητή, όπως και ισχύει στην περίπτωση μας. Για την πολυεπεξεργασία χρησιμοποιούμε την βιβλιοθήκη *joblib*, όπου και εισάγουμε την κλάση *Parallel*. Η κλάση αυτή χρησιμοποιείται για παράλληλους βρόγχους for. Ως παραμέτρους θέτουμε τον αριθμό των εργασιών (jobs), όπου στην περίπτωση μας είναι ο αριθμός των ΚΜΕ και κατόπιν την συνάρτηση delayed με το όνομα της συνάρτησης που θα χρησιμοποιήσει την πολυεπεξεργασία και τις παραμέτρους που αυτή δέχεται ως πλειάδα (tuple). Η συνάρτηση delayed είναι ένα απλό κόλπο για να μπορέσουμε να δημιουργήσουμε μια πλειάδα (*tuple*) (συνάρτηση, args, kwargs) με σύνταξη κλήσης συνάρτησης. Το σώμα της συνάρτησης export_csvs παραμένει το ίδιο, χωρίς βέβαια την παρουσία του βρόγχου *for record in records*, καθώς χρησιμοποιείται ως κριτήριο παραλληλισμού..

Κώδικας 5.2: Απόσπασμα ανάκτησης και μετατροπής δεδομένων με πολυεπεξεργασία (*multiprocessing*)

```
import multiprocessing
from joblib import Parallel, delayed

def export_csvs(record, data, data_files, dir):
    # ...

def export_csv_multiprocessing(records, data, data_files, dir):
    num_cores = multiprocessing.cpu_count()
    Parallel(num_cores - 1)(
        delayed(export_csvs)(record, data, data_files, project_dir)
        for record in records
    )
```

Η χρήση του παραπάνω κώδικα οδήγησε σε εμφανώς καλύτερα αποτελέσματα χρόνου εκτέλεσης. Για την επιβεβαίωση της υπόθεσής μας πραγματοποιήσαμε πολλαπλά πειράματα. Χρησιμοποιήσαμε διαφορετικό πλήθος αρχείων ως είσοδο και μετρήσαμε την απόδοση του κώδικα σε σχέση με το χρόνο εκτέλεσης. Τα αποτελέσματα των πειραμάτων απεικονίζονται στο σχήμα 5.3.

Όπως παρατηρούμε στο παραπάνω σχήμα, η χρήση πολυεπεξεργασίας δίνει πολύ καλύτερη απόδοση εκτέλεσης ιδιαίτερα όταν το πλήθος των αρχείων δεδομένων είναι πολύ μεγάλο. Στο πείραμα χρησιμοποιήθηκαν 12 ΚΜΕ, οπότε τα αποτελέσματα εξαρτώνται σημαντικά από αυτήν την παράμετρο. Με την αύξηση του πλήθους των αρχείων δεδομένων, ο σειριακός κώδικας Python φαίνεται να εμφανίζει όλο και μεγαλύτερους χρόνους εκτέλεσης, ενώ με την πολυεπεξεργασία ο χρόνος δεν αυξάνεται σημαντικά. Με αυτόν τον τρόπο λοιπόν, βελτιστοποιήσαμε τον κώδικα όσον αφορά τον χρόνο εκτέλεσης, γεγονός που οδηγεί και σε καλύτερη εμπειρία χρήστη



Σχήμα 5.3: Απόδοση του κώδικα ανάκτησης και μετατροπής δεδομένων

.

Κεφάλαιο 6ο: Συμπεράσματα - Προτάσεις Βελτίωσης

Στην παρούσα συζητήσαμε την σχεδίαση, ανάπτυξη και υλοποίηση ενός συστήματος ανάλυσης δεδομένων, με σεβασμό στην ιδιωτικότητα των χρηστών. Το σύστημα αυτό ήταν εμπνευσμένο από τις προκλήσεις που αντιμετωπίζουν πολλοί μικρομεσαίοι οργανισμοί στην απόκτηση, αποθήκευση και ανάλυση προσωπικών δεδομένων λόγω των κανονισμών προστασίας δεδομένων. Αναλύσαμε το πρόβλημα σε μια σειρά από ερευνητικά ερωτήματα που αντιμετωπίσαμε σε αυτήν την εργασία. Συζητάμε καθένα από τα ερευνητικά ερωτήματα και την προσέγγισή μας για την αντιμετώπισή του στη συνέχεια.

- *Ποια είναι τα βασικά χαρακτηριστικά ενός συστήματος ανάλυσης δεδομένων με σεβασμό στο απόρρητο των χρηστών της;*

  Περιγράψαμε ένα σενάριο ώστε να παραθέσουμε τις απαιτήσεις για την ασφαλή ανάλυση δεδομένων με σεβασμό στο απόρρητο και να απαριθμήσουμε μια σειρά λειτουργικών και μη λειτουργικών απαιτήσεων για την προσέγγισή μας που παρατίθενται στον πίνακα 5.2. Τα αποτελέσματα αυτά είναι μέρος της διαδικασίας αναγνώρισης και αναπροσαρμογής των απαιτήσεων που βασίζονται στα πρότυπα απορρήτου καθώς και σε περιπτώσεις χρήσης από διάφορες εταιρίες.

- *Ποια είναι τα βασικά στοιχεία μιας αρχιτεκτονικής ενός συστήματος ανάλυσης δεδομένων με σεβασμό στο απόρρητο των χρηστών της;*

  Προτείναμε μια νέα προσέγγιση πλατφόρμας, της οποίας η αρχιτεκτονική παρέχει ασφαλή ανάλυση δεδομένων με τη δυνατότητα βασικού ελέγχου απορρήτου. Το σύστημα παρέχει ασφαλή αποθήκευση για τα ευαίσθητα δεδομένα των χρηστών, συνοδευόμενα από ρητή συναίνεση. Επίσης, δίνει τη δυνατότητα σε αναλυτές/ερευνητές να επιλέξουν δεδομένα προς ανάλυση για την έρευνά τους μέσω μιας διεπαφής χρήστη. Αυτές οι λειτουργίες είναι ενσωματωμένες στο βασικό στοιχείο ΑΑΔ. Ακόμη παρέχεται ένα αξιόπιστο περιβάλλον ανάλυσης για την εκτέλεση διεργασιών ανάλυσης δεδομένων με σεβασμό στο απόρρητο.

- *Πώς μπορούν να χρησιμοποιηθούν τεχνολογίες για να υλοποιηθεί η αρχιτεκτονική ενός συστήματος ανάλυσης δεδομένων με σεβασμό στην ιδιωτικότητα των χρηστών της;*

  Παραθέσαμε ορισμένες τεχνολογίες τις οποίες χρησιμοποιήσαμε για να δημιουργήσουμε το πρωτότυπο αυτού του συστήματος. Συνοδεύοντας τα κομμάτια της υλοποίησης με αποσπάσματα κώδικα, αποδείξαμε ότι ένα τέτοιο σύστημα είναι δυνατό να δημιουργηθεί και να αναπτυχθεί περαιτέρω.

Συνοψίζοντας, στα κεφάλαια υλοποίησης και αξιολόγησης, αποδείξαμε ότι η προσέγγισή μας είναι ορθή και υλοποιήσιμη. Στο κεφάλαιο αξιολόγησης παρείχαμε τρεις πραγματικές ΠΧ στον ιατρικό τομέα. Δείξαμε ότι το πρωτότυπο της πλατφόρμας είναι σε θέση να παρακολουθεί ολόκληρο τον κύκλο ζωής δεδομένων, ξεκινώντας από την μεταφόρτωση δεδομένων έως την παραγωγή αποτελεσμάτων σε μια επιστημονική μελέτη. Έτσι, η αρχιτεκτονική παρέχει μέσα για να ελεγχθεί ποια δεδομένα χρησιμοποιήθηκαν, εάν δόθηκε η συγκατάθεση ή ποιες συγκεκριμένες βιβλιοθήκες και κώδικας

χρησιμοποιήθηκαν για την παραγωγή των αποτελεσμάτων. Αυτό, με τη σειρά του, αυξάνει την *αναπαραγωγιμότητα (reproducibility)* και την εμπιστοσύνη στα αποτελέσματα. Ως εκ τούτου η υπόθεση για την σχεδίαση και υλοποίηση ενός τέτοιου συστήματος γίνεται αποδεκτή.

Όσον αφορά τις προτάσεις βελτίωσης: σε θεωρητικό επίπεδο, η υιοθέτηση της προτεινόμενης πλατφόρμας επικεντρώνεται σε τομείς εφαρμογών στους οποίους η δυνατότητα ελέγχου και το απόρρητο είναι τα βασικά ζητήματα. Αυτό θα απαιτήσει αλλαγές στο μοντέλο δεδομένων και μεταδεδομένων που χρησιμοποιείται στο ΑΑΔ και ανάπτυξη μεθόδων για την χρήση και αυτοματοποίηση παραγωγής γράφων γνώσης *(knowledge graphs)* από νέους τύπους δεδομένων.

Ένα άλλο κομμάτι της μελλοντικής εργασίας είναι η διερεύνηση της δυνατότητας υιοθέτησης τεχνολογιών Σημασιολογικού Ιστού για το ΑΠΑΔ. Στην κοινότητα του Σημασιολογικού Ιστού έχουν εμφανιστεί πρόσφατες εργασίες προς αυτή την κατεύθυνση, π.χ. σχετικά με την επεξεργασία ερωτημάτων με γνώμονα το απόρρητο. Επιπλέον, επικεντρωνόμαστε στη βελτίωση της ποιότητας του ΑΠΑΔ. Αυτό περιλαμβάνει την υιοθέτηση σύνδεσης αρχείων για τη διατήρηση του απορρήτου ώστε να βελτιωθεί η ανάλυση δεδομένων που προέρχονται από διαφορετικές εταιρείες, καθώς και την επέκταση του φάσματος των λειτουργιών του πακέτου DataSHIELD με νέους τύπους ανάλυσης, τους οποίους μπορεί να εκτελέσει ένας αναλυτής δεδομένων. Στο [50] παρατίθεται μια βελτιωμένη αρχιτεκτονική και υλοποίηση του παρόντος πρωτοτύπου.

Σε πρακτικό επίπεδο, το σύστημα θα μπορούσε να ενσωματώσει την διεπαφή δημιουργίας πειραμάτων ως μια υπηρεσία μαζί με το ΑΑΔ. Αυτός ο συνδυασμός ενδέχεται να είναι καλύτερος, καθώς η βιβλιοθήκη Invenio, προσφέρει πολλές και διάφορες λειτουργίες έτοιμες για χρήση (π.χ. εξατομικευμένα στυλ για διεπαφές χρήστη). Σημαντικό παράδειγμα αποτελούν οι διαδρομές ΔΠΕ, που μπορούν να προσαρμοστούν εύκολα στις εκάστοτε ανάγκες. Στην περίπτωση του πρωτοτύπου μας, η διεπαφή χρήστη καθώς και οι λειτουργίες μετατροπής δεδομένων, μπορούν να ενσωματωθούν στο ΑΑΔ. Αυτό βέβαια, προϋποθέτει ένα προχωρημένο επίπεδο κατανόησης των υπηρεσιών ιστού και των μικρουπηρεσιών από μέρους των προγραμματιστών, επειδή η βιβλιοθήκη παρέχει κυρίως προηγμένες λειτουργίες. Φυσικά, εάν κάποιος προβεί σε συγχώνευση των υπηρεσιών αυτών, θα πρέπει να λάβει και τα αντίστοιχα μέτρα προστασίας για ανάγνωση και εγγραφή δεδομένων και μεταδεδομένων, καθώς η προεπιλεγμένη λειτουργία της βιβλιοθήκης είναι η παραχώρηση αυτών των δικαιωμάτων σε κάθε χρήστη.

Τέλος, η απόδοση του κώδικα της ΥΡΠ για την ανάκτηση και τον μετασχηματισμό των αρχείων δεδομένων από το ΑΑΔ μπορεί να υποστεί περαιτέρω αλλαγές, γεγονός που μπορεί να οδηγήσει σε ακόμα καλύτερα αποτελέσματα, όταν το πλήθος των αρχείων δεδομένων είναι της τάξης των χιλιάδων. Χρησιμοποιήσαμε πολυεπεξεργασία με παράμετρο τον αριθμό των πυρήνων. Μια ακόμα πιθανώς καλύτερη λύση είναι ο *πολυνηματισμός* (*multithreading*). Ο πολυνηματισμός είναι μια τεχνική που εκχωρεί πολλαπλά τμήματα κώδικα σε μια ενιαία διαδικασία. Αυτά τα τμήματα κώδικα, ονομαζόμενα και ως νήματα, τρέχουν ταυτόχρονα και παράλληλα μεταξύ τους. Αυτά τα νήματα μοιράζονται τον ίδιο χώρο μνήμης σε μια διαδικασία, γεγονός που εξοικονομεί μνήμη συστήματος, αυξάνει την ταχύτητα υπολογισμού και βελτιώνει την απόδοση της εφαρμογής. Η χρήση περισσότερων νημάτων μπορεί να προσφέρει (τονίζοντας το πιθανώς) καλύτερη απόδοση.

Η επιτυχία υλοποίησης του πρωτοτύπου απέδειξε την εγκυρότητα της προτεινόμενης προσέγγισης στην υπόθεση του τρίτου ερευνητικού μας ερωτήματος και αποτελεί οδηγό για περαιτέρω υλοποιήσεις πάνω σε παρόμοια ερευνητικά ερωτήματα. Η υλοποίησή μας μπορεί να θεωρηθεί και ως ένα *Ελάχιστο Βιώσιμο Προϊόν* (*Minimum Viable Product*). Το MVP είναι μια έκδοση προϊόντος με αρκετά

χαρακτηριστικά ώστε να μπορεί να χρησιμοποιηθεί από πρώτους πελάτες, οι οποίοι στη συνέχεια μπορούν να παρέχουν σχόλια για τη μελλοντική ανάπτυξή του.

Εν κατακλείδι, δείξαμε πώς μπορεί να βελτιωθεί η *Εύρεση*, η *Προσβασιμότητα*, η *Διαλειτουργικότητα* και η *Επαναχρησιμοποίηση* των προσωπικών δεδομένων παρέχοντας μια πλατφόρμα για ανάλυση δεδομένων διατήρησης της ιδιωτικότητας που υποστηρίζει τη δυνατότητα ελέγχου. Δώσαμε ιδιαίτερη έμφαση στην προσβασιμότητα και την επαναχρησιμοποίηση. Επιτρέψαμε την προσβασιμότητα παρέχοντας ένα ελεγχόμενο περιβάλλον που επιτρέπει την ανάλυση μη τροποποιημένων προσωπικών δεδομένων, σύμφωνα με τη συγκατάθεσή του και χωρίς διαρροή προσωπικών πληροφοριών. Για να υποστηρίξουμε την επαναχρησιμοποίηση, συλλέγουμε πληροφορίες προέλευσης σε κάθε στάδιο της ανάλυσης δεδομένων, ξεκινώντας από την αναζήτηση δεδομένων έως την παραγωγή του αποτελέσματος. Οι πληροφορίες περιλαμβάνουν λεπτομέρειες για το περιβάλλον εκτέλεσης για την υποστήριξη της αναπαραγωγιμότητας. Έτσι, παρέχουμε οδηγίες σχετικά με το πώς κάποιος μπορεί να ενεργοποιήσει την πρόσβαση, να υποστηρίξει την επαναχρησιμοποίηση προσωπικών δεδομένων και να αξιοποιήσει τις δυνατότητές τους σε νέα πειράματα ηλεκτρονικής επιστήμης (eScience).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Parliament and C. of European Union (2016), ”Regulation (EU) 2016/679”, *Official Journal of the European Union*, pp. 1-88, 2016.

[2] F. D. McSherry, “Privacy integrated queries,” in Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, Jun. 29, 2009.

[3] R. Mendes and J. P. Vilela, "Privacy-Preserving Data Mining: Methods, Metrics, and Applications," in IEEE Access, vol. 5, pp. 10562-10582, 2017.

[4] A. Gaye et al., “DataSHIELD: taking the analysis to the data, not the data to the analysis,” International Journal of Epidemiology, vol. 43, no. 6. Oxford University Press (OUP), pp. 1929–1944, Sep. 27, 2014.

[5] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing,” ACM Computing Surveys, vol. 42, no. 4. Association for Computing Machinery (ACM), pp. 1–53, Jun. 2010.

[6] P. Samarati, “Protecting respondents identities in microdata release,” IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 6. Institute of Electrical and Electronics Engineers (IEEE), pp. 1010–1027, 2001.

[7] F. Prasser and F. Kohlmayer, “Putting Statistical Disclosure Control into Practice: The ARX Data Anonymization Tool,” Medical Data Privacy Handbook. Springer International Publishing, pp. 111–148, 2015.

[8] C. Dwork, “Differential Privacy: A Survey of Results,” Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–19.

[9] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, “GUPT,” Proceedings of the 2012 international conference on Management of Data - SIGMOD ’12. ACM Press, 2012.

[10] F. D. McSherry, “Privacy integrated queries,” Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. ACM, Jun. 29, 2009.

[11] I. Roy, S.T.V. Setty, A. Kilzer, V. Shmatikov and E. Witchel, Airavat: Security and Privacy for MapReduce, in: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI’10, USENIX Association, USA, 2010.

[12] S.M. Bellovin, P.K. Dutta and N. Reitinger, Privacy and Synthetic Datasets, Stan. Tech. L. Rev. 22 (2019), 1.

[13] N. Patki, R. Wedge, and K. Veeramachaneni, “The Synthetic Data Vault,” 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE, Oct. 2016.

[14] M. Weise and A. Rauber, “Open Source Secure Data Infrastructure and Processes for Data Visiting,” Zenodo, Mar. 2021.

[15] U.H.D. Alliance, Trusted Research Environments – A Strategy to Build Public Trust and Meet Changing Health Data Science Needs, 2020. https://ukhealthdata.org/wp-content/uploads/2020/07/200723-Alliance-Board_Paper-E_TRE-Green-Paper.pdf.

[16] M. Koscina, D. Manset, C. Negri Ribalta, and O. Perez, “Enabling trust in healthcare data exchange with a federated blockchain-based architecture,” IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume. ACM, Oct. 14, 2019.

[17] N. Popper, F. Endel, R. Mayer, M. Bicher, and B. Glock, “Planning Future Health: Developing Big Data and System Modelling Pipelines for Health System Research,” SNE Simulation Notes Europe, vol. 27, no. 4. ARGESIM Arbeitsgemeinschaft Simulation News, pp. 203–208, Dec. 2017.

[18] K. W. Carter et al., “ViPAR: a software platform for the Virtual Pooling and Analysis of Research Data,” International Journal of Epidemiology, vol. 45, no. 2. Oxford University Press (OUP), pp. 408–416, Oct. 08, 2015.

[19] D. Doiron et al., “Data harmonization and federated analysis of population-based studies: the BioSHaRE project,” Emerging Themes in Epidemiology, vol. 10, no. 1. Springer Science and Business Media LLC, Nov. 21, 2013.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson and B.A.y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, A. Singh and J. Zhu, eds, Proceedings of Machine Learning Research, Vol. 54, PMLR, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html.

[21] J. Sun, X. Yang, Y. Yao, A. Zhang, W. Gao, J. Xie and C. Wang, Vertical Federated Learning without Revealing Intersection Membership, 2021. https://arxiv.org/abs/2106.05508.

[22] T. Oinn et al., “Taverna: a tool for the composition and enactment of bioinformatics workflows,” Bioinformatics, vol. 20, no. 17. Oxford University Press (OUP), pp. 3045–3054, Jun. 16, 2004.

[23] K. Belhajjame, R. B’Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker et al., Prov-DM: The prov data model, W3C Recommendation (2013).

[24] T. Miksa and A. Rauber, “Using ontologies for verification and validation of workflow-based experiments,” Journal of Web Semantics, vol. 43. Elsevier BV, pp. 25–45, Mar. 2017.

[25] B. Gosswein, T. Miksa, A. Rauber, and W. Wagner, “Data Identification and Process Monitoring for Reproducible Earth Observation Research,” 2019 15th International Conference on eScience (eScience). IEEE, Sep. 2019.

[26] R. Mayer, G. Antunes, A. Caetano, M. Bakhshandeh, A. Rauber, and J. Borbinha, “Using ontologies to capture the semantics of a (business) process for digital preservation,” International Journal on Digital Libraries, vol. 15, no. 2–4. Springer Science and Business Media LLC, pp. 129–152, Mar. 04, 2015.

[27] R. Kumar and R. Goyal, “On cloud security requirements, threats, vulnerabilities and countermeasures: A survey,” Computer Science Review, vol. 33. Elsevier BV, pp. 1–48, Aug. 2019.

[28] A. M. Ellison, E. R. Boose, B. S. Lerner, E. Fong, and M. Seltzer, “The End-to-End Provenance Project,” Patterns, vol. 1, no. 2. Elsevier BV, p. 100016, May 2020.

[29] T. D. Huynh and L. Moreau, “ProvStore: A Public Provenance Repository,” Lecture Notes in Computer Science. Springer International Publishing, pp. 275–277, 2015.

[30] B. E. Ujcich, A. Bates, and W. H. Sanders, “A Provenance Model for the European Union General Data Protection Regulation,” Lecture Notes in Computer Science. Springer International Publishing, pp. 45–57, 2018.

[31] J. F. Pimentel, J. Freire, L. Murta, and V. Braganholo, “A Survey on Collecting, Managing, and Analyzing Provenance from Scripts,” ACM Computing Surveys, vol. 52, no. 3. Association for Computing Machinery (ACM), pp. 1–38, May 31, 2020.

[32] H.J. Pandit and D. Lewis, Modelling Provenance for GDPR Compliance using Linked Open Data Vocabularies., in: PrivOn@ Workshop co-located with ISWC 2017, 2017, pp. 1–15. http://ceur-ws.org/Vol-1951/PrivOn2017_paper_6.pdf.

[33] I.C. Office, Getting Ready for the GDPR, 2020. https://ico.org.uk/for-organisations/data-protection-self-assessment/.

[34] M.T. Center, Detailed GDPR Assessment, 2017. http://aka.ms/gdprdetailedassessment.

[35] “TrustArc - GDPR Compliance Toolkit”. [Online]. Available: https://info.trustarc.com/Web-Resource-2019-01-19-Nymity-GDPR-Compliance-Toolkit_LP.html.

[36] “GConsent - A consent ontology based on the GDPR”. [Online]. Available: http://openscience.adaptcentre.ie/ontologies/GConsent/docs/ontology.

[37] “Business Process Re-engineering and functional toolkit for GDPR compliance”. [Online]. Available: https://www.bpr4gdpr.eu.

[38] “Compliance Ontology”. [Online]. Available: https://www.bpr4gdpr.eu/wp-content/uploads/2019/06/D3.1-Compliance-Ontology-1.0.pdf.

[39] A. Oltramari et al., “PrivOnto: A semantic framework for the analysis of privacy policies,” Semantic Web, vol. 9, no. 2. IOS Press, pp. 185–203, Jan. 24, 2018.

[40] “FAIR Principles - Go FAIR”. [Online]. Available: https://www.go-fair.org/fair-principles/.

[41] M. D. Wilkinson et al., “The FAIR Guiding Principles for scientific data management and stewardship”, Scientific Data, vol. 3, no. 1. Springer Science and Business Media LLC, Mar. 15, 2016.

[42] “Invenio Framework”. [Online]. Available: https://inveniosoftware.org/products/framework/.

[43] J. Chan, R. Chung and J. Huang, “Python API Development Fundamentals: Develop a full-stack web application with Python and Flask”, Packt Publishing, 2019.

[44] C. Hillar, Gaston, “Hands-On RESTful Python Web Services: Develop RESTful web services or APIs with modern Python 3.7, 2nd Edition”, Packt Publishing, 2018.

[45] M. Grinberg,”The New And Improved Flask Mega-Tutorial”, Independently published, 2017.

[46] “Flask Documentation”. [Online]. Available: https://flask.palletsprojects.com/.

[47] “Opal”. [Online]. Available: https://www.obiba.org/pages/products/opal/.

[48] “FHIR: Fast Healthcare Interoperability Resources”. [Online]. Available: http://hl7.org/implement/standards/fhir/.

[49] “LOINC Codes”. [Online]. Available: https://www.hl7.org/fhir/valueset-observation-codes.html.

[50] F. Ekaputra et al., "Semantic-enabled architecture for auditable privacy-preserving data analysis", *Semantic Web. IOS Press*, pp. 1-34, Jan. 10, 2022.