ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη web εφαρμογής για την αυτοματοποιημένη Εξαγωγή Πληροφοριών και την αναγνώριση οντοτήτων από αδόμητα δεδομένα, την ενίσχυση των πληροφοριών αυτών και την παραγωγή αναφορών»

Των φοιτητών
Θεοδώρου Χρήστου 05/2023
Παναγιωτίδη Ιωάννη 08/2022

Επιβλέπων
Αντώνιος Ευσταθίου
Βαθμίδα Καθηγητής

Ημερομηνία 10-09-2025

Τίτλος Δ.Ε. “Ανάπτυξη WEB εφαρμογής για την αυτοματοποιημένη Εξαγωγή Πληροφοριών και την Αναγνώριση Οντοτήτων από Αδόμητα Δεδομένα, την Ενίσχυση των Πληροφοριών αυτών και την παραγωγή Αναφορών”

Κωδικός Δ.Ε. 25105

Ονοματεπώνυμο φοιτητή/τών
ΘΕΟΔΩΡΟΥ ΧΡΗΣΤΟΥ
ΙΩΑΝΝΗ ΠΑΝΑΓΙΩΤΙΔΗ
Ονοματεπώνυμο εισηγητή ΕΥΣΤΑΘΙΟΥ ΑΝΤΩΝΙΟΣ

Ημερομηνία ανάληψης Δ.Ε. 16-01-2025
Ημερομηνία περάτωσης Δ.Ε. 10-09-2025

Βεβαιώνω ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιέρωση»

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των σπουδών μας στο Μεταπτυχιακό Πρόγραμμα σπουδών του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Θεσσαλονίκης, ως επιστέγασμα της θεωρητικής και πρακτικής γνώσης που αποκτήθηκε κατά τη διάρκεια της φοίτησης. Αντικείμενό της αποτελεί η μελέτη, ο σχεδιασμός και η υλοποίηση μιας διαδικτυακής εφαρμογής για την αυτοματοποιημένη επεξεργασία, εξαγωγή και ενίσχυση πληροφορίας από δεδομένα, που απαντά στις σύγχρονες ανάγκες επεξεργασίας ψηφιακών ή ψηφιοποιημένων εγγράφων.

Η ιδέα αυτής της εφαρμογής γεννήθηκε από την ανάγκη διαχείρισης μεγάλου όγκου δεδομένων, τα οποία άλλες φορές έχουν γνωστή και προκαθορισμένη μορφή, δομημένη ή ημιδομημένη όπου οι εργασίες που πρέπει να πραγματοποιηθούν είναι επαναλαμβανόμενες και τετριμμένες ενώ άλλες φορές είναι τελείως αδόμητα και πρωτότυπα. Η εφαρμογή που αναπτύσσεται αποσκοπεί στην ενίσχυση της αποδοτικότητας των χρηστών, την αυτοματοποιημένη αξιοποίηση, ανάλυσή και ενίσχυση των δεδομένων αυτών, την μετατροπή τους σε αξιοποιήσιμες πληροφορίες και την εξαγωγή χρήσιμων συμπερασμάτων.

Μέσω αυτής της εργασίας είχαμε την ευκαιρία να εμβαθύνουμε από κοινού σε τεχνολογίες αιχμής, καθώς και να εφαρμόσουμε πρακτικά έννοιες που διδαχθήκαμε κατά τη διάρκεια των σπουδών μας. Παράλληλα, αναπτύξαμε δεξιότητες στον σχεδιασμό συστημάτων, στην αρχιτεκτονική λογισμικού και στη διασύνδεση πολυεπίπεδων υπηρεσιών και τεχνολογιών. Η εμπειρία αυτή μας εξόπλισε με πολύτιμα εφόδια για την επαγγελματική μας σταδιοδρομία και μας ενίσχυσε με ολοκληρωμένη τεχνογνωσία στην ανάπτυξη καινοτόμων πληροφοριακών συστημάτων.

Περίληψη

Η παρούσα διπλωματική εργασία αφορά την ανάπτυξη μιας διαδικτυακής εφαρμογής για την αυτοματοποιημένη εξαγωγή και ενίσχυση πληροφορίας, με στόχο την παραγωγή αξιοποιήσιμων αναφορών. Ανταποκρινόμενη στην ανάγκη διαχείρισης μεγάλου όγκου εγγράφων, η εφαρμογή καλείται να υποστηρίξει τόσο δομημένες όσο και αδόμητες εισροές δεδομένων, μετατρέποντάς τις σε αξιοποιήσιμες πληροφορίες.

Η μεθοδολογία περιλαμβάνει ανάλυση απαιτήσεων, σχεδίαση της αρχιτεκτονικής και επιλογή κατάλληλων τεχνολογιών. Το σύστημα βασίζεται στο framework Django (Python) για την ενορχήστρωση των υπηρεσιών του back-end και την απόδοση μέρος του front-end, ενώ χρησιμοποιείται Nginx και Daphne για ασύγχρονη, ασφαλή και αποδοτική εξυπηρέτηση αιτημάτων. Η αποθήκευση αρχείων υλοποιείται με MinIO (S3 συμβατότητα), διασφαλίζοντας την εμπιστευτικότητα των δεδομένων και την επεκτασιμότητα της υποδομής.

Η επεξεργασία των αρχείων περιλαμβάνει την εξαγωγή κειμένου και μεταδεδομένων με Apache Tika και αναγνώριση χαρακτήρων (Optical Character Recognition – OCR), ενώ η αναγνώριση οντοτήτων (NER) επιτυγχάνεται μέσω λεξικών, κανόνων Regex και τεχνικών NLP με τη βιβλιοθήκη spaCy. Επίσης παρέχεται η δυνατότητα εμπλουτισμού των δεδομένων με εξωτερικές πηγές όπως Domain WHOIS και IP lookups.

Οι πληροφορίες αποθηκεύονται σε PostgreSQL, Neo4j και Qdrant ενώ αναζητούνται αποδοτικά μέσω Opensearch (Elasticsearch) με υποστήριξη λεξικής αναζήτησης (keyword search), σημασιολογικής αναζήτησης (semantic search) και υβριδικής αναζήτησης. Η εφαρμογή προσφέρει προκαθορισμένες αναφορές σε φυσική γλώσσα και αναφορές συσχέτισης (correlation reports) με οπτικοποίηση μέσω γραφημάτων.

Η αρχιτεκτονική υποστηρίζει ασύγχρονες διεργασίες με Celery και Redis, ενώ όλο το σύστημα είναι containerized μέσω Docker. Η τελική λύση αποτελεί ένα ευέλικτο, επεκτάσιμο και πλήρως αυτοματοποιημένο εργαλείο ανάλυσης εγγράφων για πολλαπλές εφαρμογές.

Επιπλέον, η εφαρμογή επεκτείνεται με ένα διαλογικό υποσύστημα ερωταποκρίσεων (CHAT) που αξιοποιεί Large Language Models (LLMs) μέσω της πλατφόρμας OLLAMA, επιτρέποντας την παραγωγή απαντήσεων σε φυσική γλώσσα βάσει των αποθηκευμένων δεδομένων. Η αρχιτεκτονική αυτού του υποσυστήματος βασίζεται στο πρότυπο Retrieval-Augmented Generation (RAG), συνδυάζοντας αναζήτηση τεκμηρίων σε Opensearch και Qdrant με την παραγωγή φυσικής γλώσσας από τοπικά εκτελούμενα LLMs. Η λύση αυτή προσφέρει αυτονομία, προστασία δεδομένων και αυξημένη ευχρηστία σε περιβάλλοντα χωρίς εξωτερικές εξαρτήσεις.

«Development of a WEB application for the automated Extraction of Information and the Identification of Entities from Unstructured Data, the Enhancement of this Information and the production of Reports»

«Christos Theodorou»

«Ioannis Panagiotidis»

Abstract

This thesis focuses on the development of a web application designed to address modern needs in processing large volumes of digital or digitized documents. The idea behind the system stems from the increasing demand to manage diverse data sources — some structured or semi-structured with repetitive patterns, others completely unstructured and unique. The proposed solution aims to enhance user efficiency through automated data analysis, enrichment, and the transformation of raw content into actionable knowledge.

The methodology involves requirements analysis, architectural design, and careful selection of appropriate technologies. The application is built on Django (Python), supporting both frontend and backend components, and is served via Nginx and Gunicorn for optimized, secure request handling. Uploaded files are stored using the object storage platform MinIO, ensuring data confidentiality and future scalability.

Documents undergo automated extraction via Apache Tika and OCR, followed by Named Entity Recognition (NER) using dictionaries, Regex rules, and NLP pipelines powered by the spaCy library. Users can validate or correct extracted entities and enhance them using open external data sources, including WHOIS APIs and blockchain explorers.

Processed information is stored in PostgreSQL and indexed in Elasticsearch to enable high-performance search with autocomplete. The application generates two types of reports: predefined analytical summaries and correlation reports identifying common entities between documents or cases. These relations are visualized using graph structures powered by Neo4j, allowing analysts to detect meaningful links.

To handle computationally intensive tasks asynchronously, the system utilizes Celery and Redis, while the entire architecture is containerized via Docker for consistent deployment across environments. The result is a fully modular, scalable, and automated platform that empowers users to analyze unstructured data with accuracy and depth.

Additionally, the application is expanding with a conversational question-answering subsystem (CHAT) that leverages Large Language Models (LLMs) through the OLLAMA platform, allowing the generation of natural language responses based on stored data. The architecture of this module is based on the Retrieval-Augmented Generation (RAG) standard, combining document search in Opensearch and Qdrant with natural language generation from locally executed LLMs. This solution provides autonomy, data protection, and increased usability in environments without external dependencies.

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή και βαθιά μου ευγνωμοσύνη στον επιβλέποντα καθηγητή μου, κ. Αντωνίου Ευστάθιο, για την πολύτιμη επιστημονική καθοδήγηση, τη διαρκή του υποστήριξη και την εμπιστοσύνη που μου έδειξε καθ' όλη τη διάρκεια της διπλωματικής εργασίας.

Ένα μεγάλο ευχαριστώ από καρδιάς στον φίλο και συνάδελφο Θεοδώρου Χρήστο, με τον οποίο μοιραστήκαμε όλη αυτή τη διαδρομή. Η συνεργασία μας, η αλληλοϋποστήριξη και το κοινό μας πείσμα να ξεπεράσουμε κάθε δυσκολία, έδωσαν πραγματικό νόημα σε αυτή την προσπάθεια.

Τέλος, ευχαριστώ τα παιδιά μου Αρετή, Κασσιανή, Δημήτρη, Εμμέλεια και τον Μάρκο για τον χρόνο που τους στέρησα και σύζυγό μου Μαρίνα για την υπομονή της.

Ιωάννης Παναγιωτίδης

Θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Ευστάθιο Αντωνίου, για την καθοριστική συμβολή του στην ολοκλήρωση της παρούσας διπλωματικής εργασίας καθώς η επιστημονική γνώση, εμπειρία και καθοδήγηση αποτέλεσαν σταθερό σημείο αναφοράς σε κάθε στάδιο της εργασίας.

Θερμές ευχαριστίες οφείλω επίσης στον φίλο μου και συνεργάτη, Ιωάννη Παναγιωτίδη, για την αλληλοεκτίμηση και τη συνεχή στήριξη καθ' όλη τη διάρκεια αυτής της απαιτητικής πορείας αλλά και όλους τους συναδέλφους και φίλους για τη συνεισφορά τους στον καθορισμό των απαιτήσεων.

Τέλος, ευχαριστώ τη Μαρία και την Ειρήνη, τα παιδιά μου, για την υπομονή τους και την οικογένειά μου για τη στήριξή τους.

Χρήστος Θεοδώρου

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
API	Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface)
Apache Tika	Εργαλείο για αυτόματη εξαγωγή μεταδεδομένων και περιεχομένου κειμένου
Celery	Εργαλείο για την εκτέλεση και διαχείριση ασύγχρονων εργασιών στην Python
CLI	Διεπαφή Γραμμής Εντολών (Command Line Interface)
Elasticsearch	Εργαλείο για αποθήκευση, αναζήτηση και ανάλυση μεγάλων όγκων δεδομένων
FastAPI	Σύγχρονο web framework σε Python για ανάπτυξη APIs
Flower	Εργαλείο παρακολούθησης ουρών εργασιών Celery
GUI	Γραφικό Περιβάλλον Χρήστη (Graphical User Interface)
OLLAMA	Εργαλείο τοπικής εκτέλεσης μεγάλων γλωσσικών μοντέλων
LLM	Μεγάλα Γλωσσικά Μοντέλα (Large Language Model)
OCR	Οπτική Αναγνώριση Χαρακτήρων (Optical Character Recognition)
OpenSearch	Ανοικτού κώδικα μηχανή αναζήτησης και analytics, συμβατή με Elasticsearch
NLP	Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing)
NER	Αναγνώριση Οντοτήτων (Named Entity Recognition)
MinIO	Σύστημα αποθήκευσης αντικειμένων υψηλών επιδόσεων, συμβατό με S3
MIME	Τύπος Πολυμέσων (Multipurpose Internet Mail Extensions)
Redis	In-memory βάση δεδομένων τύπου key–value (Remote Dictionary Server)
S3	Simple Storage Service
JSON	Μορφή Ανταλλαγής Δεδομένων (JavaScript Object Notation)
JS	Javascript
PostgreSQL	Ανοικτού κώδικα σχεσιακή βάση δεδομένων υψηλών δυνατοτήτων
Regex	Regular Expressions
RAG	Retrieval-Augmented Generation
spaCy	Library for NLP in Python

Κεφάλαιο 1ο: Περιγραφή παρούσας κατάστασης

1.1 Εισαγωγή

Η κατανόηση της παρούσας κατάστασης στον χώρο των λογισμικών και των οργανισμών που διαχειρίζονται πληροφορίες αποτελεί βασικό βήμα για τη διαμόρφωση αποτελεσματικών λύσεων. Στην παρούσα ενότητα επιχειρείται μια συνολική αποτύπωση της παρούσας πραγματικότητας, εστιάζοντας τόσο στις τεχνολογικές επιλογές που κυριαρχούν στην αγορά όσο και στις λειτουργικές προκλήσεις που αντιμετωπίζουν οι οργανισμοί. Η ανάλυση αυτή επιτρέπει την αναγνώριση των αδυναμιών και των περιορισμών που επηρεάζουν την αποδοτικότητα και την ευελιξία των συστημάτων διαχείρισης πληροφορίας, θέτοντας τις βάσεις για τον σχεδιασμό και την υλοποίηση πιο καινοτόμων, προσαρμοστικών και οικονομικά αποδοτικών λύσεων.

1.2 Παρούσα Κατάσταση στα λογισμικά Διαχείρισης Πληροφοριών

Η σύγχρονη διαχείριση πληροφοριών εξελίσσεται συνεχώς, ωστόσο, παρά την τεχνολογική πρόοδο, παραμένουν σημαντικά εμπόδια στη λειτουργική υιοθέτηση και αξιοποίηση των διαθέσιμων εργαλείων. Ολοένα και περισσότερες υπηρεσίες και οργανισμοί επενδύουν σε εξειδικευμένα λογισμικά με στόχο τη συλλογή, ανάλυση και αξιολόγηση πληροφοριών, ιδιαίτερα στον τομέα της ασφάλειας και ειδικότερα στην πρόληψη της απάτης και της διαφθοράς.
Ωστόσο, τα υπάρχοντα λογισμικά που κυριαρχούν στην αγορά, παρουσιάζουν συγκεκριμένα χαρακτηριστικά και περιορισμούς:

- **Άδεια Χρήσης & Κόστος**: Τα περισσότερα από αυτά τα εργαλεία είναι **εμπορικά προϊόντα (licensed software)**, με σημαντικό κόστος **απόκτησης, συντήρησης και ανανέωσης**. Το κόστος αυτό συχνά λειτουργεί ως αποτρεπτικός παράγοντας για μικρούς οργανισμούς ή οργανισμούς με μικρή χρηματοδότηση.
- **Έλλειψη Παραμετροποίησης**: Πολλά από τα εργαλεία αυτά είναι **κλειστά συστήματα** που δεν επιτρέπουν σημαντική παραμετροποίηση ή προσαρμογή στις ιδιαίτερες ανάγκες του κάθε οργανισμού, περιορίζοντας έτσι την ευελιξία και τη χρηστικότητα τους σε συγκεκριμένα σενάρια.
- **Απαιτήσεις σε Πόρους**: Η λειτουργία τους απαιτεί σημαντικούς **υπολογιστικούς πόρους από την πλευρά του τελικού χρήστη (client-side)**, ιδιαίτερα όσον αφορά τη μνήμη, την ταχύτητα επεξεργασίας και την ανάγκη για συνεχή συνδεσιμότητα με βάσεις δεδομένων ή APIs για την πλήρη αξιοποίηση των δυνατοτήτων τους.
- **Απαίτηση Εξειδικευμένων Γνώσεων**: Η λειτουργία και αξιοποίηση των παραπάνω εργαλείων προϋποθέτει **εξειδικευμένη τεχνογνωσία**, τόσο σε επίπεδο ανάλυσης δεδομένων όσο και στον χειρισμό των ίδιων των εργαλείων. Η καμπύλη εκμάθησης είναι απότομη, γεγονός που δυσκολεύει τη χρήση από **μη τεχνικούς αναλυτές ή προσωπικό χωρίς προηγούμενη εμπειρία στον τομέα**.
- **Περιορισμένο Υποστηρικτικό Υλικό (Support)**: Παρά την πολυπλοκότητα τους, τα παραδοσιακά λογισμικά **συνοδεύονται από περιορισμένη τεχνική υποστήριξη ή μη εκτενή τεκμηρίωση**. Σε ορισμένες περιπτώσεις, η υποστήριξη παρέχεται μόνο για συγκεκριμένα χρονικά διαστήματα ή απαιτεί πρόσθετες χρεώσεις, γεγονός που οδηγεί σε καθυστερήσεις κατά την ενσωμάτωσή τους σε παραγωγικά περιβάλλοντα.
- **Αργή Εξέλιξη και Περιορισμένες Αναβαθμίσεις:** Ένα σημαντικό πρόβλημα αφορά την **έλλειψη συνεχούς αναβάθμισης**. Μετά την αρχική πώληση και εγκατάσταση, **δεν υφίσταται**

ενεργή συναλλακτική σχέση μεταξύ παρόχου και πελάτη, με αποτέλεσμα οι ενημερώσεις να διατίθενται με το αργά και όχι σύμφωνα με τις πραγματικές ανάγκες των χρηστών.

- **Συνδρομητικές Υπηρεσίες (SaaS):** Σύγχρονα εργαλεία που βασίζονται σε cloud πλατφόρμες και ακολουθούν το μοντέλο «λογισμικό ως υπηρεσία» (Software as a Service - SaaS), προσφέρουν **συνεχή ενημέρωση και τεχνική υποστήριξη**, συχνά βάσει του **feedback των χρηστών**. Ωστόσο, η χρήση τους προϋποθέτει **συνεχή καταβολή συνδρομής**, γεγονός που μπορεί να οδηγήσει σε **σωρευτικό κόστος** μακροπρόθεσμα.

Αυτοί οι παράγοντες δημιουργούν την ανάγκη για πιο **προσβάσιμα, παραμετροποιήσιμα και αποδοτικά εργαλεία διαχείρισης πληροφοριών**, τα οποία θα μπορούν να προσαρμόζονται σε διαφορετικά περιβάλλοντα και χρήστες, χωρίς να απαιτούν εξειδικευμένη τεχνογνωσία ή υπερβολικούς πόρους.

1.3 Ενδεικτικά Παραδείγματα Εφαρμογών

Στο πλαίσιο αυτής της εργασίας, παρουσιάζονται ενδεικτικά λογισμικά/εργαλεία που χρησιμοποιούνται ή μπορούν να χρησιμοποιηθούν για τη συλλογή, επεξεργασία και ανάλυση πληροφοριών. Η επιλογή των παραδειγμάτων βασίζεται κυρίως σε **προσωπική εμπειρία χρήσης** και παρατήρηση της λειτουργικότητάς τους στο πλαίσιο συγκεκριμένων επαγγελματικών ή εκπαιδευτικών σεναρίων και όχι από ενδελεχή τεχνική ή εμπορική ανάλυση.

Η παρούσα αποτίμηση αφορά αποκλειστικά ορισμένες εκδόσεις των επιλεγμένων λογισμικών (π.χ. desktop client) και **δεν επεκτείνεται στο σύνολο των διαθέσιμων παραλλαγών κάθε προϊόντος**, όπως enterprise, web-based ή cloud λύσεις. Επομένως, τα συμπεράσματα δεν θα πρέπει να εκληφθούν ως καθολικά για την κάθε οικογένεια προϊόντων.

1.3.1 Maltego

Το **Maltego** είναι ένα από τα πλέον διαδεδομένα εργαλεία ανάλυσης πληροφοριών και χρησιμοποιείται κυρίως για την **εισαγωγή, συλλογή, οπτικοποίηση και ανάλυση δεδομένων** από ανοιχτές ή κλειστές πηγές πληροφόρησης, με σημαντική εφαρμογή στον χώρο του **OSINT (Open Source Intelligence)**.
Παρά τις ισχυρές δυνατότητες του εργαλείου, παρουσιάζει τεχνικούς και λειτουργικούς περιορισμούς που το καθιστούν **κατάλληλο μόνο για συγκεκριμένα σενάρια χρήσης**.

Πλεονεκτήματα

- **Μεγάλη και ενεργή βάση χρηστών**: Το Maltego έχει αναπτυχθεί από το 2007 και χρησιμοποιείται παγκοσμίως από αναλυτές πληροφοριών, δημοσιογράφους, αστυνομικές αρχές και υπηρεσίες ασφαλείας, κάτι που συνεπάγεται μεγάλη τεκμηρίωση, κοινότητες υποστήριξης και ευρεία διάδοση.
- **Ισχυρή οπτικοποίηση σχέσεων**: Παρέχει διαδραστικά γραφήματα (graphs) με κόμβους και ακμές που απεικονίζουν σχέσεις μεταξύ οντοτήτων με ευκρίνεια και ευελιξία.
- **Ενσωμάτωση με εξωτερικές OSINT υπηρεσίες**: Μέσω του μηχανισμού των **Transformations**, μπορεί να αντλήσει πληροφορίες από DNS records, WHOIS, social media, Shodan, VirusTotal κ.ά.
- **Διαθέσιμο σε διαφορετικές εκδόσεις**: Παρέχει **δωρεάν έκδοση (Community Edition)** για εκπαιδευτική ή ελαφριά χρήση, και επαγγελματικές εκδόσεις με περισσότερες δυνατότητες.

Μειονεκτήματα

- **Επεξεργάζεται μόνο δομημένα δεδομένα**: Δεν υποστηρίζει την ανάλυση αδόμητων δεδομένων (π.χ. πλήρες κείμενο αναφορών, emails, αρχεία Word ή PDF) χωρίς εξωτερική επεξεργασία.

- **Απαιτεί χειροκίνητο mapping**: Για την εισαγωγή αρχείων (CSV, XLSX κ.ά.) ο χρήστης πρέπει να αντιστοιχίσει τα πεδία του αρχείου στα entities του Maltego, διαδικασία που **δεν είναι αυτοματοποιημένη** και απαιτεί τεχνική εξοικείωση.
- **Υψηλές απαιτήσεις σε πόρους και γνώση**: Η χρήση του σε μεγαλύτερα projects απαιτεί σημαντική **υπολογιστική ισχύ και εμπειρία**, καθώς και καλή γνώση του πλαισίου των OSINT τεχνικών.
- **Περιορισμοί στις δωρεάν εκδόσεις**: Πολλά Transforms και δυνατότητες απαιτούν **επιπλέον άδειες χρήσης ή συνδρομές** σε τρίτους παρόχους υπηρεσιών.
- **Ζητήματα ιδιωτικότητας**: Τα δεδομένα που υποβάλλονται για επεξεργασία μέσω cloud (όπως τα Transforms) ενδέχεται να αποσταλούν εκτός του ελεγχόμενου περιβάλλοντος του χρήστη, γεγονός που μπορεί να προκαλέσει ανησυχίες σε ευαίσθητα σενάρια.

1.3.2 Tovek

Το **Tovek** είναι ένα εργαλείο που σχεδιάστηκε για την αναζήτηση, επεξεργασία και ανάλυση **μεγάλου όγκου αδόμητων δεδομένων**, όπως κείμενα ελεύθερης μορφής, αναφορές, email, έγγραφα PDF και άλλα αρχεία χωρίς προκαθορισμένη δομή. Αν και μπορεί να φανεί εξαιρετικά χρήσιμο σε περιβάλλοντα όπου η ταχύτητα επεξεργασίας είναι κρίσιμη, εμφανίζει σημαντικούς περιορισμούς ως προς τη **διαφάνεια, την παραμετροποίηση και την εννοιολογική ανάλυση** των δεδομένων.

Πλεονεκτήματα

- **Υψηλή ταχύτητα επεξεργασίας**: Το Tovek ξεχωρίζει για τη δυνατότητά του να επεξεργάζεται γρήγορα μεγάλους όγκους εγγράφων, γεγονός που το καθιστά ιδανικό για **μαζική προ-ανάλυση** ή **ταχεία επισκόπηση μεγάλων αρχείων**.
- **Κατάλληλο για αδόμητα δεδομένα**: Εστιάζει στην αναγνώριση πληροφορίας από μη δομημένα κείμενα, κάτι που αποτελεί πλεονέκτημα σε σύγκριση με εργαλεία που περιορίζονται σε structured input.
- **Αυτοματοποιημένη αναγνώριση**: Χρησιμοποιεί λεξικά και regex για εντοπισμό βασικών οντοτήτων (ονόματα, τοποθεσίες, αριθμούς, ημερομηνίες), χωρίς να απαιτεί ενεργή παρέμβαση από τον χρήστη.

Μειονεκτήματα

- **"Black box" λειτουργία**: Ο χρήστης **δεν έχει πρόσβαση ούτε έλεγχο στους κανόνες** που εφαρμόζονται για την αναγνώριση των οντοτήτων. Δεν υποστηρίζεται η δημιουργία προσαρμοσμένων κανόνων ή η τροποποίηση των υπαρχόντων.
- **Περιορισμένη ακρίβεια αναγνώρισης**: Η ανάλυση βασίζεται σε **στατικά λεξικά** και **κανονικές εκφράσεις (regex)**, με αποτέλεσμα να μην εντοπίζονται όλες οι πιθανές οντότητες — ειδικά όταν η γλώσσα του κειμένου αποκλίνει από προβλεπόμενες μορφές.
- **Αδυναμία διορθώσεων από τον χρήστη**: Το εργαλείο **δεν παρέχει δυνατότητα επανεπεξεργασίας ή διόρθωσης** λανθασμένα ταυτοποιημένων οντοτήτων, γεγονός που περιορίζει την αξιοπιστία των αποτελεσμάτων σε ευαίσθητα ή σύνθετα σενάρια.
- **Απλουστευμένη εννοιολογική ανάλυση**: Το μόνο είδος σχέσης που εντοπίζεται είναι η **συνεμφάνιση οντοτήτων στο ίδιο έγγραφο**. Δεν αναγνωρίζονται σημασιολογικές ή λογικές συσχετίσεις μεταξύ των οντοτήτων (π.χ. ποιος ανήκει σε ποιον, ποιος κάνει τι).
- **Περιορισμένη επεκτασιμότητα**: Δεν υποστηρίζει ενσωμάτωση με άλλα εργαλεία ή pipelines, ούτε διαθέτει API-based προσβασιμότητα.

1.3.3 IBM i2 Analyst's Notebook

Το **IBM i2 Analyst's Notebook** αποτελεί ένα από τα πλέον εξειδικευμένα εργαλεία για **ανάλυση σχέσεων και οπτικοποίηση συνδέσεων**. Χρησιμοποιείται ευρέως από υπηρεσίες ασφαλείας, διωκτικές αρχές, αναλυτές οικονομικού εγκλήματος και ερευνητικά κέντρα. Σκοπός του είναι η **ανάδειξη**

συσχετίσεων, προτύπων και διασυνδέσεων μεταξύ οντοτήτων, μέσα από την επεξεργασία και απεικόνιση δομημένων δεδομένων.

Πλεονεκτήματα

- **Πλήρης υποστήριξη δομημένων δεδομένων**: Το i2 εισάγει δεδομένα από αρχεία Excel, CSV και βάσεις δεδομένων, επιτρέποντας την εύκολη δημιουργία αναλυτικών γραφημάτων και τη μοντελοποίηση σύνθετων οντοτήτων.
- **Εξελιγμένη οπτικοποίηση σχέσεων**: Παρέχει διαγράμματα σχέσεων (link charts) με **χρονική διάσταση, ιεραρχία και κατηγοριοποίηση**, επιτρέποντας την εις βάθος διερεύνηση σχέσεων και ακολουθιών ενεργειών.
- **Ρητός καθορισμός σχέσεων**: Ο χρήστης μπορεί να **καθορίσει με ακρίβεια το είδος της σχέσης** (π.χ. ιδιοκτησία, επικοινωνία, μεταφορά χρημάτων), προσφέροντας περισσότερη νοηματική πληροφορία σε σχέση με εργαλεία που βασίζονται απλώς σε συνεμφάνιση (όπως το Tovek).
- **Δυνατότητα ενσωμάτωσης με άλλα εργαλεία και βάσεις**: Συνδέεται με πλατφόρμες όπως **iBase, ESRI GIS, SQL βάσεις**, γεγονός που το καθιστά ιδανικό για πολυεπίπεδες επιχειρησιακές αναλύσεις.

Μειονεκτήματα

- **Υψηλό κόστος**: Το λογισμικό είναι **εμπορικό με υψηλή τιμή κτήσης**, ενώ απαιτείται και **πρόσθετη άδεια για συνοδευτικά εργαλεία** (π.χ. iBase, chart templates, training modules).
- **Απαίτηση εξειδίκευσης**: Η χρήση του απαιτεί **ειδική εκπαίδευση**, καθώς δεν είναι εύχρηστο για αρχάριους ή μη τεχνικούς αναλυτές. Η καμπύλη εκμάθησης είναι απότομη, ειδικά σε περιβάλλοντα χωρίς υπάρχουσα υποδομή.
- **Δεν υποστηρίζει αυτόνομα αδόμητα δεδομένα**: Για να εξάγει πληροφόρηση από ελεύθερο κείμενο (π.χ. έγγραφα, αναφορές, emails), χρειάζεται **προεπεξεργασία με άλλα εργαλεία** όπως text mining engines ή εξωτερικές NLP λύσεις.
- **Απουσία άμεσης σύνδεσης με ανοιχτές πηγές (OSINT)**: Δεν διαθέτει ενσωματωμένους μηχανισμούς online αναζήτησης ή επαφής με APIs τρίτων (σε αντίθεση με το Maltego). Τα δεδομένα πρέπει να έχουν συγκεντρωθεί και καθαριστεί πριν την ανάλυση.

1.3.4 KNIME

Το **KNIME (Konstanz Information Miner)** είναι ένα **ανοιχτού κώδικα λογισμικό** για την οπτική μοντελοποίηση ροών εργασίας στην επεξεργασία και ανάλυση δεδομένων. Αναπτύχθηκε αρχικά από το Πανεπιστήμιο της Konstanz και σήμερα χρησιμοποιείται ευρέως σε επιχειρήσεις, ακαδημαϊκά ιδρύματα και ερευνητικά εργαστήρια.

Πλεονεκτήματα

- **Δωρεάν και επεκτάσιμο**: Η βασική έκδοση του KNIME είναι **εντελώς δωρεάν** (KNIME Analytics Platform) και υποστηρίζει ένα **ευρύ οικοσύστημα από nodes, επεκτάσεις και integrations** με εργαλεία τρίτων (Python, R, Spark, TensorFlow, Hugging Face NLP κ.ά.).
- **Οπτική ροή εργασίας (no-code/low-code)**: Επιτρέπει στον χρήστη να σχεδιάζει ροές ανάλυσης δεδομένων με **διαισθητικό interface**, χωρίς να απαιτείται προγραμματισμός, μέσω «μπλοκ» (nodes) που συνδέονται μεταξύ τους.
- **Υποστήριξη αδόμητων και δομημένων δεδομένων**: Μπορεί να επεξεργάζεται **κείμενα, email, PDF, XML, JSON**, καθώς και δομημένα δεδομένα από Excel, βάσεις δεδομένων ή APIs.

- **Ενεργή κοινότητα και συνεχή ανάπτυξη**: Διαθέτει **μεγάλη κοινότητα χρηστών**, συνεχή ενημέρωση, tutorials, φόρουμ και δωρεάν διαθέσιμα workflows.

Μειονεκτήματα

- **Μονοχρηστική λειτουργία**: Η δωρεάν έκδοση (KNIME Analytics Platform) **δεν υποστηρίζει πολλαπλούς χρήστες** ή ταυτόχρονη συνεργασία. Κάθε χρήστης εργάζεται **τοπικά και μεμονωμένα**. Για υποστήριξη πολλαπλών χρηστών, απαιτείται η συνδρομητική έκδοση (**KNIME Business Hub / Server**), η οποία παρέχει συνεργατικές δυνατότητες και web-based πρόσβαση.
- **Δεν είναι εξειδικευμένο εργαλείο OSINT ή ανάλυσης σχέσεων**: Δεν προσφέρει ενσωματωμένα modules για συλλογή πληροφοριών από ανοιχτές πηγές ή για σύνθετη οπτικοποίηση σχέσεων όπως το Maltego ή το i2. Αυτές οι δυνατότητες πρέπει να κατασκευαστούν από τον χρήστη με πρόσθετα εργαλεία ή προγραμματισμό.
- **Απαίτηση βασικών τεχνικών γνώσεων**: Η αποτελεσματική χρήση του KNIME απαιτεί κατανόηση βασικών εννοιών δεδομένων, προεπεξεργασίας και ροής εργασίας. Η πρώτη επαφή μπορεί να αποδειχθεί αποθαρρυντική για εντελώς αρχάριους.

1.4 Παρούσα Κατάσταση στους Οργανισμούς

Παρά τη διαθεσιμότητα τεχνολογικών λύσεων, η καθημερινή διαχείριση πληροφορίας στους οργανισμούς παραμένει κατακερματισμένη, μη αυτοματοποιημένη και εξαρτώμενη από χειρωνακτικές διαδικασίες. Τα κυριότερα προβλήματα εντοπίζονται στα εξής σημεία:

- **Κατακερματισμός της πληροφορίας:** Η πληροφορία είναι διάσπαρτη σε διαφορετικά **συστήματα, υπηρεσίες, αρχεία, τμήματα και μορφές** (π.χ. Excel, PDF, email, reports), χωρίς ενιαίο σημείο ενοποίησης ή **διαλειτουργικότητας**. Αυτό δημιουργεί πολλαπλές εκδοχές των ίδιων δεδομένων και καθυστερεί τη λήψη τεκμηριωμένων αποφάσεων.
- **Απουσία υποδομών και συστημάτων ολοκλήρωσης:** Οι περισσότεροι οργανισμοί **δεν διαθέτουν τεχνικές υποδομές** (servers, cloud περιβάλλον, APIs, εργαλεία διασύνδεσης) ικανές να συγκεντρώσουν και να αξιοποιήσουν τη διαθέσιμη πληροφορία με ασφαλή και αποτελεσματικό τρόπο.
- **Πληθώρα και πολυμορφία πληροφορίας:** Οι οργανισμοί καλούνται να διαχειριστούν **μεγάλο όγκο πληροφορίας** που μπορεί να είναι: **Δομημένη** (π.χ. πίνακες Excel, βάσεις δεδομένων), **Ημιδομημένη** (π.χ. email headers, αρχεία XML/JSON) ή **Αδόμητη** (π.χ. κείμενα, εκθέσεις, έγγραφα PDF, αποφάσεις) ενώ ταυτόχρονα, **λείπουν τα εργαλεία που να μπορούν να διαχειριστούν οριζόντια όλα αυτά τα επίπεδα πληροφορίας**.
- **Επαναληψιμότητα ενεργειών σε ίδια ή παρόμοια έγγραφα:** Συχνά, **οι ίδιες ή παρόμοιες εργασίες εκτελούνται χειροκίνητα** σε πανομοιότυπα έγγραφα (π.χ. πολλαπλές αναζητήσεις, αντιγραφή στοιχείων, εξαγωγή και ανάλυση βασικών δεδομένων). Αυτές οι διαδικασίες είναι χρονοβόρες, **επιρρεπείς σε λάθη** και **δύσκολα ελέγξιμες**.
- **Έλλειψη εκπαιδευμένου προσωπικού και τεχνικής υποστήριξης:** Η τεχνολογία συχνά δεν συνοδεύεται από κατάλληλη επιμόρφωση ή τεχνική υποστήριξη. Το αποτέλεσμα είναι η **υποχρησιμοποίηση διαθέσιμων εργαλείων**, καθώς και η **αδυναμία διάγνωσης ή επίλυσης προβλημάτων** σε πραγματικό χρόνο.

1.5 Επίλογος

Από την παρούσα αποτίμηση προκύπτει ότι, παρά την πληθώρα εργαλείων και τεχνολογικών υποδομών, η διαχείριση πληροφορίας παραμένει συχνά κατακερματισμένη, ακριβή και περιορισμένη σε δυνατότητες προσαρμογής. Οι οργανισμοί έρχονται αντιμέτωποι με προβλήματα που αφορούν το υψηλό κόστος, την έλλειψη παραμετροποίησης, την ανάγκη εξειδικευμένης τεχνογνωσίας και την απουσία ολοκληρωμένων υποδομών. Ταυτόχρονα, οι υπάρχουσες πλατφόρμες συχνά αδυνατούν να αξιοποιήσουν πλήρως τα δεδομένα που διαχειρίζονται, περιορίζοντας την επιχειρησιακή ευελιξία και την ικανότητα λήψης τεκμηριωμένων αποφάσεων. Τα συμπεράσματα αυτά υπογραμμίζουν την ανάγκη για νέα μοντέλα και εργαλεία που θα προσφέρουν ολοκληρωμένες, ασφαλείς και εύχρηστες λύσεις, ανοίγοντας τον δρόμο για την ανάλυση των σύγχρονων τεχνολογικών προσεγγίσεων στο επόμενο κεφάλαιο.

Κεφάλαιο 2ο: Καθορισμός αναγκών / σχεδιασμός λύσης

2.1 Εισαγωγή

Αφού εξετάστηκαν ορισμένες από τις διαθέσιμες εμπορικές λύσεις για τη συλλογή, ανάλυση και διαχείριση πληροφοριών, αναδείχθηκαν σημαντικά πλεονεκτήματα, αλλά και κρίσιμοι λειτουργικοί και τεχνικοί περιορισμοί και διαμορφώθηκε η ανάγκη για την ανάπτυξη μίας **εσωτερικής λύσης**, προσαρμοσμένης σε **πραγματικές ανάγκες.**

Παρά την προσπάθεια ισορροπίας μεταξύ τεχνικής αξιοπιστίας, χρηστικότητας και επιχειρησιακής ευελιξίας, αναγνωρίζεται ότι η εφαρμογή **δεν είναι πλήρως απαλλαγμένη από αδυναμίες**. Ωστόσο, προσφέρει ένα ρεαλιστικό, παραμετροποιήσιμο και επεκτάσιμο σημείο αναφοράς για την **περαιτέρω ανάπτυξη και βελτίωση στο συγκεκριμένο πλαίσιο** χρήσης.

2.2 Λειτουργικές Απαιτήσεις της Εφαρμογής

Οι λειτουργικές απαιτήσεις της εφαρμογής αφορούν τις επιμέρους δυνατότητες και διεργασίες που πρέπει να υποστηρίζει το σύστημα για να καλύπτει τις ανάγκες των χρηστών και του οργανωτικού πλαισίου στο οποίο εντάσσεται. Αντί να παρουσιαστούν αποσπασματικά, οι απαιτήσεις αυτές περιγράφονται αναλυτικά στις επόμενες ενότητες, στο πλαίσιο της παρουσίασης της λειτουργικότητας και των ροών εργασίας της εφαρμογής.

Η επιλογή αυτή επιτρέπει την ενσωμάτωσή τους στο κατάλληλο λειτουργικό και τεχνικό πλαίσιο, προσφέροντας έτσι πληρέστερη κατανόηση του τρόπου με τον οποίο ικανοποιούνται στην πράξη.

2.3 Μη Λειτουργικές Απαιτήσεις της Εφαρμογής

Οι μη λειτουργικές απαιτήσεις καθορίζουν τις ποιοτικές ιδιότητες της εφαρμογής και συμβάλλουν καθοριστικά στη βιωσιμότητα, χρηστικότητα και τεχνική ωριμότητά της. Οι βασικές απαιτήσεις που τέθηκαν κατά τον σχεδιασμό είναι οι εξής:

- **Απαιτήσεις απόδοσης:** Η εφαρμογή πρέπει να μπορεί να επεξεργάζεται **εκατοντάδες** έγγραφα ημερησίως, με δεκάδες ταυτόχρονους χρήστες, χωρίς καθυστερήσεις. Ο μέσος χρόνος απόκρισης για βασικές ενέργειες χρήστη (π.χ. προβολή, αποθήκευση, αναζήτηση στοιχείου) δεν πρέπει να υπερβαίνει τα 2 δευτερόλεπτα. Οι διεργασίες υψηλής υπολογιστικής απαίτησης πρέπει να εκτελούνται ασύγχρονα στο παρασκήνιο, χωρίς να διακόπτεται η ροή εργασίας του χρήστη.
- **Απαιτήσεις Ευχρηστίας:** Η διεπαφή της εφαρμογής πρέπει να είναι απλή, καθοδηγούμενη και προσαρμοσμένη στο προφίλ μη τεχνικών χρηστών. Η ροή εργασιών ακολουθεί ξεκάθαρα, προκαθορισμένα βήματα, ενώ κάθε χρήστης έχει πρόσβαση μόνο στις λειτουργίες που αντιστοιχούν στον ρόλο του. Επιπλέον, η διεπαφή πρέπει να παραμένει πλήρως λειτουργική ακόμη και σε τερματικά χαμηλών προδιαγραφών, χωρίς να απαιτείται εγκατάσταση πρόσθετου λογισμικού.
- **Απαιτήσεις Αξιοπιστίας:** Το σύστημα πρέπει να συνεχίζει να λειτουργεί ακόμα και αν μία ή περισσότερες επιμέρους υπηρεσίες αποτύχουν. Η αρχιτεκτονική της εφαρμογής επιτρέπει την απομόνωση σφαλμάτων, καταγραφή σφαλμάτων (logging) και δυνατότητα ανάκτησης

κατάστασης. Τα αρχεία και τα δεδομένα πρέπει να διατηρούνται ακέραια, ακόμα και σε περίπτωση απώλειας σύνδεσης ή διακοπής λειτουργίας.

- **Απαιτήσεις Ασφαλείας:** Η πρόσβαση στο σύστημα επιτρέπεται μόνο μέσω πιστοποιημένης αυθεντικοποίησης χρηστών. Υποστηρίζεται ο διαχωρισμός δικαιωμάτων πρόσβασης ανά ρόλο και οργανωτική μονάδα. Τα δεδομένα μεταφέρονται μέσω κρυπτογραφημένων καναλιών (HTTPS) και αποθηκεύονται με προστασία από μη εξουσιοδοτημένη πρόσβαση. Επιπλέον, οι ενέργειες των χρηστών καταγράφονται για λόγους ιχνηλασιμότητας.
- **Απαιτήσεις Συντηρησιμότητας:** Η εφαρμογή είναι σχεδιασμένη σε αρθρωτή (modular) αρχιτεκτονική, επιτρέποντας την εύκολη ενημέρωση ή αντικατάσταση μεμονωμένων υποσυστημάτων. Όλος ο πηγαίος κώδικας είναι τεκμηριωμένος, ενώ οι ρυθμίσεις και οι παράμετροι διαχωρίζονται από τον κώδικα μέσω αρχείων περιβάλλοντος. Έτσι, οι διορθώσεις και οι εξελίξεις μπορούν να γίνουν χωρίς ριζικές αλλαγές.
- **Απαιτήσεις Επεκτασιμότητας:** Η εφαρμογή πρέπει να μπορεί να επεκταθεί οριζόντια (προσθήκη νέων server/containers) και κάθετα (αύξηση υπολογιστικών πόρων), χωρίς ανασχεδιασμό. Η χρήση ασύγχρονων και διακριτών υπηρεσιών επιτρέπει την κλιμάκωση σε περιβάλλοντα αυξανόμενης χρήσης ή απαιτήσεων.
- **Απαιτήσεις Φορητότητας**: Η εφαρμογή πρέπει να μπορεί να αναπτυχθεί σε διαφορετικά περιβάλλοντα (τοπικοί υπολογιστές υψηλών δυνατοτήτων, servers, cloud υποδομές). Η χρήση τεχνολογιών containerization εξασφαλίζει ότι η εγκατάσταση και η μεταφορά της εφαρμογής γίνεται χωρίς τροποποιήσεις, υποστηρίζοντας ευκολία επανεκκίνησης και φορητότητα μεταξύ περιβαλλόντων.

2.4 Βασικά Δομικά Στοιχεία Αρχιτεκτονικής

Μετά τον προσδιορισμό των μη λειτουργικών απαιτήσεων, καθορίστηκαν τα βασικά δομικά στοιχεία της αρχιτεκτονικής του συστήματος, με κύριο στόχο την κατά το δυνατόν πλήρη ικανοποίησή τους.
Στην ενότητα που ακολουθεί, περιγράφονται οι βασικές τεχνολογικές επιλογές που υιοθετήθηκαν – όπως η χρήση τεχνολογιών containerization, η γλώσσα προγραμματισμού και η υλοποίηση της εφαρμογής ως web app – καθώς και τα πλεονεκτήματα που προσφέρουν σε σχέση με τις απαιτήσεις του συστήματος.

2.4.1 Χρήση Docker για Φορητότητα και Συμβατότητα

Η εφαρμογή υλοποιήθηκε με χρήση της τεχνολογίας Docker και οργανώθηκε σε πολλαπλά απομονωμένα containers, με στόχο τη φορητότητα, συμβατότητα και ευκολία ανάπτυξης σε ετερογενή περιβάλλοντα. Η επιλογή αυτή ανταποκρίνεται σε βασικές μη λειτουργικές απαιτήσεις, όπως η υποστήριξη εγκατάστασης, η επεκτασιμότητα και η συντηρησιμότητα, και προσφέρει τα εξής πλεονεκτήματα:

- **Πλατφορμική ανεξαρτησία (cross-platform portability)**: Η εφαρμογή μπορεί να εγκατασταθεί με τον ίδιο τρόπο σε Windows ή Linux, εξαλείφοντας ζητήματα συμβατότητας μεταξύ περιβάλλοντος ανάπτυξης και παραγωγής.
- **Απλοποιημένη ανάπτυξη και αναπαραγωγή**: Η εγκατάσταση γίνεται με μία εντολή (docker compose up), χωρίς χειροκίνητη εγκατάσταση εξαρτήσεων, ρυθμίσεων ή βάσεων δεδομένων, μειώνοντας δραστικά τον χρόνο υλοποίησης και την πιθανότητα λαθών.

- **Ευελιξία σε τοπική ή server εγκατάσταση**: Η εφαρμογή μπορεί να λειτουργήσει είτε σε προσωπικό υπολογιστή είτε σε κεντρικό server, υποστηρίζοντας ταυτόχρονα πολλούς χρήστες με την ίδια αρχιτεκτονική.
- **Απομόνωση περιβάλλοντος και αυξημένη αξιοπιστία**: Κάθε επιμέρους υπηρεσία (όπως ανάλυση κειμένου, αποθήκευση, αναζήτηση) εκτελείται σε ξεχωριστό container, διασφαλίζοντας ότι πιθανό σφάλμα δεν επηρεάζει τα υπόλοιπα συστατικά, γεγονός που ενισχύει την ανθεκτικότητα και την ευκολία συντήρησης.

Η χρήση του Docker επιτρέπει την εύκολη μεταφορά, την τυποποιημένη ανάπτυξη και την επεκτασιμότητα της εφαρμογής που επιτρέπει την εκκίνηση με μία εντολή (docker compose up),

Οι βασικές υπηρεσίες, οι οποίες αναλυτικά περιγράφονται παρακάτω, ορίζονται συγκεντρωτικά στο αρχείο **docker-compose.yml** [12] και είναι οι είναι:

- **Django (backend)** [1] – βασικό backend της εφαρμογής
- **MinIO (object storage)** [4] – διαχείριση αποθήκευσης εγγράφων
- **PostgreSQL** – σχεσιακή βάση δεδομένων
- **OpenSearch (2 nodes)** [8] – κατανεμημένη μηχανή αναζήτησης
- **Apache Tika** [5] – parsing και εξαγωγή μεταδεδομένων από έγγραφα
- **FastAPI NER API** [2] – οντολογική αναγνώριση μέσω spaCy
- **Redis** – μεσολαβητής μηνυμάτων για τις ασύγχρονες εργασίες
- **Celery Workers (3 containers)** [3], [9] – εκτέλεση παράλληλων διεργασιών
- **Neo4j** [10] – βάση γραφημάτων για αποτύπωση σχέσεων
- **Nginx** – reverse proxy για web traffic & SSL termination
- **Flower** – εργαλείο παρακολούθησης Celery queues
- **HTTrack** – custom crawler (για εισαγωγή εξωτερικής πληροφορίας)
- **External API service** – ενδιάμεσο επίπεδο για APIs
- **Qdrant** – vector database για αποθήκευση εννοιών/embeddings
- **Ollama** – lightweight LLM runtime για τοπική inference

Παράλληλα, η παραμετροποίηση της εγκατάστασης γίνεται αποκλειστικά μέσω **αρχείων περιβάλλοντος (.env)**, χωρίς ανάγκη αλλαγών στον πηγαίο κώδικα παρέχοντας κυρίως **αυξημένη ασφάλεια**, μέσω αποφυγής αποθήκευσης διαπιστευτηρίων (credentials) στον κώδικα.

Η αρχιτεκτονική περιλαμβάνει **πολλαπλά volumes** για την **επιμονή δεδομένων** (persistency) και την **κοινή χρήση πληροφορίας** μεταξύ υπηρεσιών:

- minio_data, postgres_data, neo4j_data, qdrant_data: για αποθήκευση δεδομένων ανάλογα με την κάθε τεχνολογία
- ollama_models: αποθήκευση των LLM μοντέλων που χρησιμοποιούνται τοπικά
- static_volume: φιλοξενία των στατικών αρχείων Django για σερβίρισμα μέσω Nginx

Επίσης προβλέφθηκε η μονάδα αποθήκευσης **shared_logs** που αποτελεί **κεντρικό σημείο** συγκέντρωσης αρχείων καταγραφής, από πολλές υπηρεσίες ταυτόχρονα. Η ύπαρξη αυτού του volume διευκολύνει την ενιαία παρακολούθηση της συμπεριφοράς του συστήματος, τον εντοπισμό σφαλμάτων και την ανάλυση συμβάντων (auditability), καθώς τα logs μπορούν να συλλεχθούν ή να αποσταλούν σε εξωτερικό σύστημα παρακολούθησης χωρίς αποσύνδεση της υπηρεσίας.

2.4.2 Γλώσσα Ανάπτυξης

Για την υλοποίηση του συστήματος επιλέχθηκε η γλώσσα προγραμματισμού **Python**, καθώς συνδυάζει τεχνολογική ωριμότητα με πρακτική ευελιξία, ιδιαίτερα στον τομέα της **επεξεργασίας φυσικής γλώσσας (NLP)** και διαθέτει ένα πλούσιο οικοσύστημα βιβλιοθηκών αιχμής. Παράλληλα, η

εκφραστική και λιτή σύνταξή της διευκολύνει την ανάπτυξη και μετέπειτα συντήρηση πολύπλοκων λειτουργιών.
Επιπλέον, η Python παρέχει εξαιρετικές δυνατότητες διασύνδεσης με άλλες τεχνολογίες, καθώς υποστηρίζει τη χρήση βάσεων δεδομένων, RESTful APIs, μηχανών αναζήτησης και διαφόρων άλλων υπηρεσιών. Αυτή η διαλειτουργικότητα επιτρέπει στο σύστημα να λειτουργεί ως ενορχηστρωτής της πληροφορίας, ενισχύοντας τη δυνατότητα μελλοντικής επέκτασης. Η συγκεκριμένη επιλογή συνδέεται άμεσα με κρίσιμες μη λειτουργικές απαιτήσεις, όπως η επεκτασιμότητα, η συντηρησιμότητα και η υποστήριξη προηγμένων δυνατοτήτων επεξεργασίας δεδομένων.

2.4.3 Web-based Διεπαφή Χρήστη

Η επιλογή υλοποίησης της εφαρμογής ως web εφαρμογή, η οποία είναι προσβάσιμη μέσω **φυλλομετρητή** (**browser**), έγινε με στόχο την καλύτερη ικανοποίηση κρίσιμων μη λειτουργικών απαιτήσεων, όπως η ευκολία συντήρησης, η επεκτασιμότητα και η χρηστικότητα. Η web αρχιτεκτονική επιτρέπει τη χρήση της εφαρμογής από οποιαδήποτε συσκευή με σύνδεση στο δίκτυο, χωρίς την ανάγκη εγκατάστασης τοπικού λογισμικού, διευκολύνοντας έτσι την καθολική διάθεση σε οργανισμούς με πολλαπλά τμήματα ή γεωγραφικά διασκορπισμένες δομές.
Παράλληλα, η κεντρική διαχείριση του κώδικα και των ρυθμίσεων περιορίζει σημαντικά την πολυπλοκότητα συντήρησης, καθώς οι αναβαθμίσεις πραγματοποιούνται μία φορά στον server. Η responsive σχεδίαση της διεπαφής διασφαλίζει ομαλή εμπειρία χρήσης και από κινητές συσκευές ή tablets, ενώ η δυνατότητα λειτουργίας είτε σε τοπικό υπολογιστή είτε σε εξυπηρετητή καθιστά την εφαρμογή ευέλικτη απέναντι σε περιορισμούς υποδομών. Συνολικά, η επιλογή αυτή ενισχύει τη λειτουργική απλότητα και προσφέρει ευκολία στην καθημερινή χρήση, χωρίς να επιβάλλει τεχνικά εμπόδια στους τελικούς χρήστες.

2.4.4 Σχεδιαστικοί Περιορισμοί

Κατά τον σχεδιασμό της εφαρμογής λήφθηκαν υπόψη ορισμένοι ουσιαστικοί περιορισμοί, οι οποίοι δεν αποτέλεσαν μόνο τεχνικά εμπόδια, αλλά λειτούργησαν και ως καθοδηγητικές παράμετροι για τη λήψη κρίσιμων αποφάσεων αρχιτεκτονικής. Οι περιορισμοί αυτοί αφορούν τόσο τις υποδομές και τις συνθήκες χρήσης, όσο και τη φύση των δεδομένων και τις απαιτήσεις εμπιστευτικότητας.
Ένας βασικός περιορισμός σχετίζεται με την ανάπτυξη και χρήση της εφαρμογής σε περιβάλλοντα με περιορισμένους πόρους. Καθώς το σύστημα προορίζεται να λειτουργεί σε οργανισμούς με ετερογενείς τεχνολογικές υποδομές – ενδεχομένως ακόμη και σε προσωπικούς σταθμούς εργασίας χωρίς εξειδικευμένο εξοπλισμό – κρίθηκε απαραίτητο να επιλεγούν τεχνικές λύσεις ελαφριές, με χαμηλές απαιτήσεις σε επεξεργαστική ισχύ και μνήμη. Αυτό διασφαλίζει τη σταθερή λειτουργία της εφαρμογής χωρίς να προϋποτίθεται η ύπαρξη εξελιγμένου λογισμικού ή hardware. Ωστόσο, η αρχιτεκτονική της εφαρμογής υποστηρίζει πλήρως και την ανάπτυξη σε περιβάλλοντα υψηλότερων προδιαγραφών. Μέσω κατάλληλης παραμετροποίησης (π.χ. αύξηση των διαθέσιμων πόρων σε επιμέρους υπηρεσίες, ενεργοποίηση περισσότερων instances παράλληλης επεξεργασίας), η εφαρμογή μπορεί να κλιμακωθεί και να αξιοποιήσει επαρκώς τις επιπλέον δυνατότητες που παρέχει ένα πιο ισχυρό σύστημα. Έτσι, εξασφαλίζεται η ευελιξία εγκατάστασης τόσο σε περιορισμένα όσο και σε πιο απαιτητικά επιχειρησιακά περιβάλλοντα.
Παράλληλα, ο σχεδιασμός της εφαρμογής έπρεπε να ανταποκριθεί στην πρόκληση της επεξεργασίας ετερογενών μορφών δεδομένων. Το σύστημα καλείται να χειριστεί αρχεία διαφορετικής δομής και

προέλευσης, όπως PDF, εικόνες, έγγραφα κειμένου, πίνακες ή ακόμη και με ημιδομημένο περιεχόμενο. Η πολυμορφία αυτή συνεπάγεται αυξημένη πολυπλοκότητα ως προς την ορθή εξαγωγή πληροφορίας, την ταξινόμηση και την κανονικοποίηση δεδομένων, ιδιαίτερα όταν αυτά συνοδεύονται από γλωσσικές ιδιαιτερότητες.

Επιπλέον, η εφαρμογή αναπτύχθηκε με ιδιαίτερη μέριμνα για την προστασία της εμπιστευτικότητας των δεδομένων. Δεδομένου ότι συχνά γίνεται διαχείριση ευαίσθητης πληροφορίας, η αρχιτεκτονική βασίστηκε στην αρχή της ελαχιστοποίησης εξωτερικών εξαρτήσεων. Προτιμάται η χρήση τοπικών μοντέλων γλώσσας και εργαλείων ανάλυσης, ενώ η επικοινωνία με εξωτερικά APIs περιορίζεται απολύτως, μόνο όταν είναι τεχνικά αναγκαία. Με αυτόν τον τρόπο ενισχύεται η αυτονομία του συστήματος και μειώνεται η έκθεση σε εξωτερικούς κινδύνους.

Τέλος, η εφαρμογή σχεδιάστηκε ώστε να ενσωματώνεται ομαλά στις υπάρχουσες οργανωτικές δομές, χωρίς να απαιτεί ριζικές αλλαγές στη λειτουργική κουλτούρα ή τις καθιερωμένες διαδικασίες. Αντί να επιβάλει νέες ροές, προσαρμόζεται στις υφιστάμενες, χρησιμοποιώντας ορολογία και πρότυπα κατανοητά στους χρήστες. Αυτό εξασφαλίζει υψηλή αποδοχή και μειώνει τη γραφειοκρατική ή εκπαιδευτική επιβάρυνση. Παράλληλα, η ευελιξία του συστήματος επιτρέπει την περαιτέρω προσαρμογή του σε μελλοντικές λειτουργικές απαιτήσεις, χωρίς την ανάγκη ανασχεδιασμού.

2.5 Ενσωμάτωση σε υπάρχουσες οργανωτικές δομές

Η σχεδίαση της εφαρμογής θεμελιώθηκε πάνω στην αρχή ότι το έγγραφο (**Document**) αποτελεί τον πρωταρχικό πυρήνα της πληροφορίας και τον κύριο φορέα νοήματος μέσα στο σύστημα. Η επιλογή αυτή δεν είναι τυχαία, αλλά συνάδει με τη διοικητική πρακτική όπου τα έγγραφα λειτουργούν ως επίσημα τεκμήρια και σημεία αναφοράς για κάθε διαδικασία. Η πληροφορία δεν νοείται ως αυτόνομο απόσπασμα δεδομένων, αλλά ως περιεχόμενο που τεκμηριώνεται, επεξεργάζεται και αξιολογείται με αναφορά στην αρχική της πηγή. Κατ' αυτόν τον τρόπο εξασφαλίζεται η ιχνηλασιμότητα, η δυνατότητα ελέγχου της προέλευσης και η εγκυρότητα των συμπερασμάτων.

Η ενσωμάτωση των εγγράφων στο πλαίσιο μιας υπόθεσης (**Case**) αποτυπώνει τη λειτουργική πραγματικότητα των οργανισμών, όπου οι υποθέσεις συγκροτούνται μέσα από τη συλλογή και μελέτη συσχετιζόμενων εγγράφων. Η επεξεργασία των εγγράφων (Document) οδηγεί στην ανάδειξη οντοτήτων (**Entity**), οι οποίες συνιστούν δομημένες αναπαραστάσεις κρίσιμων στοιχείων. Με τον τρόπο αυτό η αδόμητη πληροφορία μετατρέπεται σε αναλύσιμο και αξιοποιήσιμο υλικό, χωρίς ωστόσο να αποκόπτεται από την πηγή της. Η σχέση Case–Document–Entity δεν αποτελεί τεχνητό σχήμα, αλλά φυσική αναπαράσταση του τρόπου με τον οποίο οι οργανισμοί ήδη εργάζονται: η υπόθεση συγκροτείται, τα έγγραφα κατατίθενται και τα ευρήματα προκύπτουν ως απόρροια της εξέτασής τους.

Εξίσου κρίσιμη είναι η αντιστοίχιση της εφαρμογής με την οργανωτική ιεραρχία. Οι έννοιες Διεύθυνση (**Division**) και Τμήμα (**Department**) ενσωματώνονται με τρόπο που αντικατοπτρίζει τη διοικητική δομή, ενώ οι χρήστες κατανέμονται σε αυτά τα επίπεδα σύμφωνα με τους πραγματικούς τους ρόλους. Η ορατότητα των Case και Document καθορίζεται βάσει των βαθμίδων πρόσβασης που εφαρμόζονται ήδη σε οργανισμούς: **ιδιωτική (private)** χρήση σε περιορισμένο κύκλο (χρήστης και επόπτης), τοπική (local) διάθεση στο πλαίσιο τμήματος και δημόσια (public) προσβασιμότητα όταν αυτό κρίνεται αναγκαίο. Έτσι, η εφαρμογή δεν επιβάλλει νέες μορφές ιεραρχίας ούτε ανατρέπει την καθιερωμένη λογική διαχείρισης· αντίθετα, την κωδικοποιεί σε ένα ψηφιακό σχήμα με μεγαλύτερη σαφήνεια και διαφάνεια.

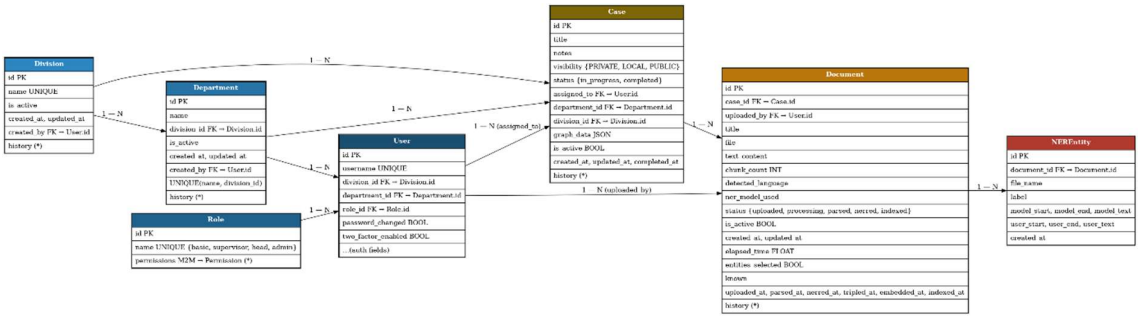
Πέραν της οργανωτικής διάρθρωσης, η εφαρμογή αναγνωρίζει και την αναγκαιότητα ρητής διαβάθμισης ρόλων. Ο **απλός χρήστης (basic)** περιορίζεται σε βασικές λειτουργίες προβολής και καταχώρισης, ο **επόπτης (supervisor)** διαθέτει διευρυμένες δυνατότητες διαχείρισης των υποθέσεων

και εγγράφων στο πλαίσιο του τμήματός του, ενώ ο **επικεφαλής (head)** έχει τη δυνατότητα παρέμβασης σε ευρύτερο επίπεδο Διεύθυνσης. Τέλος, ο **διαχειριστής (administrator)** διαθέτει πλήρη δικαιώματα επί του συστήματος. Η διαβάθμιση αυτή κατοχυρώνει την αντιστοίχιση της εφαρμογής με τα πραγματικά επίπεδα ευθύνης που υφίστανται στους οργανισμούς και ταυτόχρονα ενισχύει την ασφάλεια, τη διαφάνεια και τη λογοδοσία στη διαχείριση της πληροφορίας.

Η επιλογή αυτή ευνοεί τη λειτουργική συνέχεια και την αποδοχή από το προσωπικό, καθώς η χρήση της εφαρμογής δεν απαιτεί ριζικές αλλαγές στη γραφειοκρατική κουλτούρα ή στις τυπικές διαδικασίες. Αντί να εισάγει καινοφανείς ροές, το σύστημα προσαρμόζεται στις ήδη υπάρχουσες, χρησιμοποιώντας ορολογία οικεία και πρότυπα αναγνωρίσιμα από τους χρήστες. Παράλληλα, η δομή παραμένει επαρκώς ευέλικτη ώστε να υποστηρίζει μελλοντικές προσαρμογές και νέες λειτουργικές ανάγκες χωρίς να απαιτείται πλήρης ανασχεδιασμός.

Συνολικά, η αρχιτεκτονική της εφαρμογής μπορεί να χαρακτηριστεί ως ενσυνείδητη μεταφορά της υφιστάμενης οργανωτικής λογικής σε ψηφιακή μορφή. Με τον τρόπο αυτό επιτυγχάνεται η ομαλή ενσωμάτωση στις υπάρχουσες δομές, η ενίσχυση της εγκυρότητας των δεδομένων και η δημιουργία ενός πλαισίου που ενώνει την επιχειρησιακή πρακτική με τις τεχνολογικές δυνατότητες.

Η παραπάνω λογική αποτυπώνεται σχηματικά στη δομή της βάσης δεδομένων, όπου οι οντότητες Case, Document και Entity συνδέονται με την οργανωτική ιεραρχία και τους ρόλους των χρηστών. Το σχήμα που ακολουθεί παρουσιάζει τον τρόπο με τον οποίο η εγγραφοκεντρική λογική ενσωματώνεται στις οργανωτικές διαδικασίες, ενώ παράλληλα αναδεικνύει την εφαρμογή των κανόνων ορατότητας σύμφωνα με τις υφιστάμενες πολιτικές πρόσβασης πληροφορίας.



Σχήμα 2.1: Εννοιολογικό σχήμα της βάσης δεδομένων.

2.6 Επίλογος

Η ανάλυση των υπαρχουσών εμπορικών λύσεων και των τεχνολογικών περιορισμών που τις συνοδεύουν κατέδειξε την ανάγκη για την ανάπτυξη μιας προσαρμοσμένης, εσωτερικής λύσης, ικανής να συνδυάζει τεχνική αξιοπιστία, παραμετροποίηση και λειτουργική ευελιξία. Μέσα από τον σαφή προσδιορισμό των λειτουργικών και μη λειτουργικών απαιτήσεων, καθώς και τη στοχευμένη επιλογή τεχνολογιών, διαμορφώθηκε μια αρχιτεκτονική που ανταποκρίνεται στις ανάγκες του οργανισμού και μπορεί να κλιμακωθεί ή να τροποποιηθεί σε μελλοντικές συνθήκες.

Η επιλογή τεχνολογιών όπως το Docker για φορητότητα, η Python για ευελιξία στην ανάπτυξη, και η web-based αρχιτεκτονική για απρόσκοπτη πρόσβαση, θέτουν τις βάσεις για την υλοποίηση των επόμενων σταδίων. Το επόμενο κεφάλαιο εστιάζει στο πρώτο λειτουργικό στάδιο της εφαρμογής, το οποίο αφορά τη μεταφόρτωση εγγράφων και την αυτόματη εξαγωγή πληροφορίας, διαδικασία που

αποτελεί τον ακρογωνιαίο λίθο για τη μετέπειτα σημασιολογική ανάλυση και επεξεργασία των δεδομένων.

Κεφάλαιο 3ο: Πρώτο στάδιο ανάπτυξης – Μεταφόρτωση Εγγράφων και Εξαγωγή Δεδομένων (Document Upload & Data Extraction)

3.1 Εισαγωγή

Το πρώτο λειτουργικό στάδιο της εφαρμογής επικεντρώνεται στη μεταφόρτωση εγγράφων και στην αυτόματη εξαγωγή πληροφορίας – διαδικασία που αποτελεί τον θεμέλιο λίθο για όλα τα επόμενα στάδια ανάλυσης και επεξεργασίας. Στο σημείο αυτό, η εφαρμογή καλείται να διαχειριστεί δεδομένα ετερογενούς μορφής (όπως PDF, DOCX, εικόνες, TXT, CSV κ.ά.) και να προχωρήσει στην εξαγωγή του περιεχομένου τους, ανεξαρτήτως της αρχικής δομής ή πολυπλοκότητας.
Στο παρόν κεφάλαιο παρουσιάζονται οι βασικές και συμπληρωματικές τεχνολογίες που επιλέχθηκαν για την υλοποίηση των σχετικών διεργασιών, οι λόγοι πίσω από τις επιλογές αυτές, καθώς και ο τρόπος με τον οποίο ενσωματώνονται λειτουργικά στην εφαρμογή. Επιπλέον, περιγράφεται αναλυτικά η ροή εισαγωγής εγγράφων, τα τεχνικά στάδια της εξαγωγής δεδομένων, καθώς και οι διεπαφές που καθοδηγούν τον χρήστη κατά τη διαδικασία.

3.2 Βασικές Τεχνολογίες – Τεκμηρίωση της επιλογής τους

Η σχεδίαση και ανάπτυξη της παρούσας εφαρμογής βασίστηκε σε ένα σύνολο τεχνολογιών ανοικτού κώδικα που επιλέχθηκαν στρατηγικά με γνώμονα την επεκτασιμότητα, την ασφάλεια και την αρθρωτή αρχιτεκτονική. Στο επίκεντρο της εφαρμογής βρίσκεται το **Django**, ένα πλήρες web framework για την Python, το οποίο αξιοποιήθηκε τόσο για την ανάπτυξη του backend όσο και για τη διαχείριση των RESTful διεπαφών. Η εξαγωγή περιεχομένου από μη δομημένα και ετερογενή έγγραφα υποστηρίζεται μέσω του **Apache Tika**, που ενσωματώνει OCR δυνατότητες (μέσω του Tesseract) και παρέχει ενιαία πρόσβαση σε μεγάλο εύρος τύπων αρχείων. Για την αποθήκευση των εγγράφων χρησιμοποιήθηκε το **MinIO**, ένα σύστημα αντικειμενοστραφούς αποθήκευσης, το οποίο διασφαλίζει την ανεξαρτησία από συγκεκριμένους παρόχους cloud και επιτρέπει την ευέλικτη διαχείριση δεδομένων σε self-hosted περιβάλλοντα. Οι παραπάνω τεχνολογίες συνθέτουν τον πυρήνα του συστήματος, προσφέροντας ένα σταθερό και παραμετροποιήσιμο υπόβαθρο για την αυτοματοποιημένη επεξεργασία εγγράφων μεγάλης κλίμακας.

3.2.1 Django

Το **Django** αποτελεί ένα ευρέως διαδεδομένο και ισχυρό web framework ανοιχτού κώδικα, ανεπτυγμένο στη γλώσσα προγραμματισμού Python. Εμφανίστηκε αρχικά για την κάλυψη των απαιτήσεων ταχύτατης ανάπτυξης εφαρμογών σε δημοσιογραφικά περιβάλλοντα (newsrooms), και έκτοτε εξελίχθηκε σε πλήρες εργαλείο ανάπτυξης διαδικτυακών εφαρμογών. Χαρακτηρίζεται από υψηλό επίπεδο ασφάλειας, επεκτασιμότητας και απόδοσης [21].

Σχήμα 3.1: Λογότυπο του framework Django

Πηγή: Official Django Logos (https://www.djangoproject.com/community/logos/)

Υιοθετεί τη φιλοσοφία *"batteries-included"*, προσφέροντας ενσωματωμένα εργαλεία για τη διαχείριση βάσεων δεδομένων μέσω ORM (Object-Relational Mapping), μηχανισμούς αυθεντικοποίησης και εξουσιοδότησης, φόρμες, διαχείριση URL δρομολόγησης, middleware, καθώς και υποστήριξη RESTful APIs μέσω του **Django REST Framework (DRF)**. Επιπλέον, περιλαμβάνει κρίσιμες λειτουργίες ασφαλείας, όπως προστασία από CSRF επιθέσεις και μηχανισμούς ασφαλούς αποθήκευσης κωδικών πρόσβασης [22], [23].

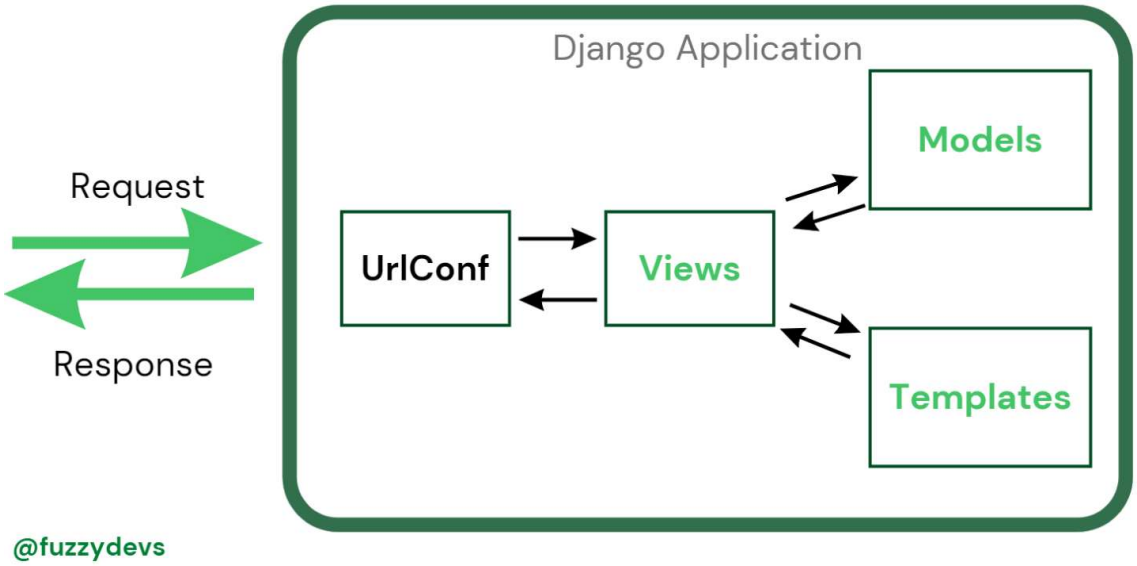
Ένα σημαντικό χαρακτηριστικό του Django είναι η ενσωματωμένη υποστήριξη για **RBAC (Role-Based Access Control)**, το οποίο επιτρέπει τη διαχείριση ρόλων, ομάδων και δικαιωμάτων με σαφήνεια και προσαρμοστικότητα [13]. Στην παρούσα εφαρμογή, η δυνατότητα αυτή αξιοποιήθηκε για την εφαρμογή διαβαθμισμένων επιπέδων πρόσβασης, προκειμένου να διασφαλιστεί η προστασία δυνητικά ευαίσθητων δεδομένων. Ο ενσωματωμένος μηχανισμός εξουσιοδότησης και το πλήρως παραμετροποιήσιμο διαχειριστικό περιβάλλον (admin interface) επιτρέπουν την εφαρμογή πολιτικών ασφαλείας χωρίς την ανάγκη χρήσης εξωτερικών μηχανισμών.

Το Django είναι πλήρως συμβατό με τις πιο διαδεδομένες σχεσιακές βάσεις δεδομένων, όπως PostgreSQL, MySQL, SQLite και Oracle. Παράλληλα, προσφέρει ευκολία ενσωμάτωσης με message queues, REST APIs και εξωτερικές υπηρεσίες. Η ευρεία υιοθέτηση του από οργανισμούς όπως η Mozilla, το Instagram, η NASA και το Pinterest καταδεικνύει τη σταθερότητα και την αξιοπιστία του [25], [26].

Ιδιαίτερης σημασίας για την παρούσα εφαρμογή είναι η υποστήριξη ασύγχρονης επεξεργασίας (*asynchronous processing*), η οποία επιτρέπει την εκτέλεση απαιτητικών διεργασιών, όπως η ανάλυση εγγράφων, η εξαγωγή οντοτήτων (NER) και ο υπολογισμός embeddings, χωρίς την επιβάρυνση της διεπαφής χρήστη. Οι διεργασίες αυτές μπορούν να εκτελούνται είτε παράλληλα είτε σε χρονικά προγραμματισμένες φάσεις (*deferred execution*), συμβάλλοντας στην απρόσκοπτη λειτουργία και στην ομαλή εμπειρία χρήστη ακόμα και υπό υψηλό φόρτο.

3.2.1.1 Django MVT

Συγκεκριμένα, η παρούσα εφαρμογή αναπτύχθηκε σύμφωνα με το αρχιτεκτονικό πρότυπο MVT (Model–View–Template), το οποίο αποτελεί παραλλαγή του ευρέως διαδεδομένου προτύπου MVC (Model–View–Controller) και είναι ειδικά σχεδιασμένο για το πλαίσιο εργασίας Django.



Σχήμα 3.2: Γραφική απεικόνιση του MVT αρχιτεκτονικού προτύπου του Django

Πηγή: Structure of Django Application (https://blog.therishabhdev.com/structure-of-django-application)

Συγκεκριμένα, η παρούσα εφαρμογή αναπτύχθηκε σύμφωνα με το αρχιτεκτονικό πρότυπο MVT (Model–View–Template), το οποίο αποτελεί παραλλαγή του ευρέως διαδεδομένου προτύπου MVC (Model–View–Controller) και είναι ειδικά σχεδιασμένο για το πλαίσιο εργασίας Django. Η αρχιτεκτονική MVT οργανώνει τη ροή ενός αιτήματος (request) και της αντίστοιχης απόκρισης (response) ως εξής:

- **Request**: Ο χρήστης αποστέλλει αίτημα προς τον διακομιστή της εφαρμογής.
- **URLconf**: Το αρχείο urls.py αντιστοιχίζει τη διαδρομή του URL στο κατάλληλο view.
- **View**: Περιέχει τη λογική της εφαρμογής. Διαχειρίζεται το εισερχόμενο αίτημα, αλληλεπιδρά με τα μοντέλα (models) και επιστρέφει την κατάλληλη απόκριση, συνήθως μέσω ενός template.
- **Model**: Ορίζει τη δομή και τις σχέσεις των δεδομένων, αξιοποιώντας το Django ORM για την πρόσβαση και διαχείριση της βάσης δεδομένων.
- **Template**: Πρόκειται για αρχεία HTML με ενσωματωμένη τη γλώσσα Django Template Language (DTL), τα οποία αποδίδουν δυναμικά το περιεχόμενο στον τελικό χρήστη.
- **Response**: Η τελική απόκριση (π.χ. HTML, JSON) αποστέλλεται πίσω στον client.

Η παραπάνω αρχιτεκτονική προάγει τον αυστηρό διαχωρισμό ευθυνών (*separation of concerns*), γεγονός που διευκολύνει σημαντικά τη συντήρηση, την επεκτασιμότητα και τη συνολική αναγνωσιμότητα του κώδικα.

Αν και το πρότυπο MVT βασίζεται σε server-side rendering, στην παρούσα εφαρμογή υιοθετήθηκε σε επιλεγμένα σημεία η απόδοση δυναμικού περιεχομένου στην πλευρά του client, με χρήση προσαρμοσμένων σεναρίων JavaScript. Μέσω της χρήσης AJAX και τεχνικών client-side rendering, επιτυγχάνεται μεγαλύτερη διαδραστικότητα και βελτιώνεται η απόκριση της διεπαφής, ιδίως σε περιπτώσεις όπου απαιτείται άμεση ενημέρωση του περιεχομένου χωρίς ανανέωση της σελίδας.

Ο συνδυασμός server-side και client-side rendering παρέχει μια ευέλικτη και αποδοτική προσέγγιση, ενισχύοντας τόσο τη σταθερότητα όσο και τη χρηστικότητα της εφαρμογής.

3.2.2 Minio

Το **Amazon S3 API (Simple Storage Service Application Programming Interface)** έχει καθιερωθεί ως το πιο ευρέως υιοθετημένο πρότυπο για την παροχή υπηρεσιών **object storage** σε σύγχρονες web εφαρμογές. Παρότι πρόκειται για τεχνολογία που αναπτύχθηκε από την Amazon Web Services (AWS), η εκτεταμένη χρήση του στην αγορά έχει καταστήσει το S3 API ένα de facto βιομηχανικό πρότυπο για τη διαχείριση και προσπέλαση αντικειμένων αποθήκευσης μέσω RESTful διεπαφών [49].

Σχήμα 3.3: Λογότυπο του Minio

Πηγή: Minio Logo Usage (https://www.min.io/compliance#Logo-Usage)

Αυτή η ευρεία αποδοχή έχει συμβάλει στην εμφάνιση εναλλακτικών, **ανοικτού κώδικα** λύσεων — με πλέον χαρακτηριστική το **MinIO** — που υλοποιούν πλήρως συμβατές διεπαφές με το S3 API, προσφέροντας παράλληλα σημαντικά πλεονεκτήματα όπως **φορητότητα, ελαφριά εγκατάσταση**, και **τεχνολογική ανεξαρτησία** από την AWS (avoiding vendor lock-in) [50].
Το S3 API υλοποιείται σύμφωνα με REST αρχιτεκτονική και υποστηρίζει HTTP μεθόδους όπως PUT (για μεταφόρτωση αντικειμένων), GET (ανάκτηση), DELETE (διαγραφή), και LIST (λίστα περιεχομένων). Παράλληλα, παρέχει δυνατότητες όπως **διαχείριση μεταδεδομένων**, **πολιτικές ελέγχου πρόσβασης**, **versioning**, και **μηχανισμούς ειδοποιήσεων (event notifications)** [51].
Στην παρούσα εφαρμογή, το **MinIO** ενσωματώθηκε ως το βασικό υποσύστημα αποθήκευσης εγγράφων, αξιοποιώντας πλήρως τις δυνατότητες του S3 API. Η επικοινωνία με το Django backend επιτυγχάνεται μέσω των Python βιβλιοθηκών boto3 και minio, διευκολύνοντας την υλοποίηση σύνθετων ροών όπως:

- **Δημιουργία presigned URLs**, για ασφαλή και προσωρινή πρόσβαση σε συγκεκριμένα αρχεία χωρίς πλήρη αυθεντικοποίηση [42][51].
- **Οργανωμένη διαχείριση buckets**, επιτρέποντας τον λογικό διαχωρισμό εγγράφων (π.χ. raw/processed/reports) με σκοπό την ιχνηλασιμότητα και την απομόνωση δεδομένων.
- **Ενεργοποίηση event-based μηχανισμών**, όπου η προσθήκη ενός νέου αντικειμένου ενεργοποιεί διαδικασίες ασύγχρονης επεξεργασίας (όπως OCR, parsing ή enrichment) μέσω Celery workers [52].

Η **Dockerized εγκατάσταση** του MinIO σε self-hosted περιβάλλον επιτρέπει την ανεξάρτητη λειτουργία της αποθήκευσης από τα υπόλοιπα υποσυστήματα, υποστηρίζοντας έτσι modular και ευέλικτες αρχιτεκτονικές με δυνατότητα εύκολης μεταφοράς μεταξύ περιβαλλόντων ανάπτυξης και παραγωγής [53].
Επιπλέον, η πλήρης **συμβατότητα με το S3 API** καθιστά εφικτή τη μελλοντική **μετεγκατάσταση** της εφαρμογής σε AWS S3 ή άλλο cloud provider, χωρίς αλλαγές στον κώδικα, αρκεί να διατηρηθεί η ίδια διεπαφή [54]. Η επιλογή αυτή προσφέρει **μακροπρόθεσμη τεχνολογική βιωσιμότητα**, περιορίζοντας την εξάρτηση από τρίτους παρόχους [41][50].
Σε αντιδιαστολή με τις παραδοσιακές file-based λύσεις, το object storage μοντέλο του MinIO επιτρέπει την αποθήκευση κάθε αρχείου ως ανεξάρτητου αντικειμένου με συνοδευτικά μεταδεδομένα,

ενισχύοντας τη **διαλειτουργικότητα**, τον **λεπτομερή έλεγχο πρόσβασης** και την **επεκτασιμότητα** της αρχιτεκτονικής [43][44].
Τέλος, για την ενίσχυση της ασφάλειας και της αξιοπιστίας των δεδομένων, αξιοποιούνται λειτουργίες όπως:

- **Encryption-at-rest και in-transit** για την προστασία κατά την αποθήκευση και τη μεταφορά [45].
- **Versioning** και **access policies** για την παρακολούθηση αλλαγών και την ενδελεχή διαχείριση προσβάσεων [52].
- **Notification hooks** για την αυτόματη διασύνδεση με εξωτερικά υποσυστήματα όπως Redis και Tika, υποστηρίζοντας event-driven αρχιτεκτονική [52].

Η επιλογή και ενσωμάτωση του MinIO συνέβαλε καθοριστικά στην επίτευξη των στόχων της εφαρμογής ως προς την **ασφάλεια, την τεχνική ανεξαρτησία, την ευελιξία ενοποίησης και την ετοιμότητα για παραγωγική λειτουργία σε cloud και on-premises υποδομές**.

3.2.3 Apache Tika

Το **Apache Tika** αποτελεί ένα framework ανοιχτού κώδικα, αναπτυγμένο από το **Apache Software Foundation**, το οποίο έχει σχεδιαστεί για την **αυτόματη αναγνώριση, ανάλυση και εξαγωγή περιεχομένου και μεταδεδομένων** από πληθώρα μορφότυπων αρχείων. Υλοποιείται τόσο ως **Java βιβλιοθήκη**, όσο και ως **αυτόνομος RESTful server** (Tika Server), διευκολύνοντας την ενσωμάτωσή του σε πολυγλωσσικά περιβάλλοντα εφαρμογών μέσω HTTP APIs [55].

Σχήμα 3.4: Λογότυπο του Apache Tika

Πηγή: Official Tika Website (https://tika.apache.org/)

Το Tika λειτουργεί ως ενδιάμεσο επίπεδο (middleware) μεταξύ των υποσυστημάτων αποθήκευσης και επεξεργασίας εγγράφων, παρέχοντας ομοιογενή διεπαφή πρόσβασης στο περιεχόμενο, ανεξαρτήτως του τύπου του αρχείου. Ο πυρήνας του αξιοποιεί υπάρχουσες βιβλιοθήκες parsing (όπως Apache POI, PDFBox, Metadata Extractor) και τις ενοποιεί κάτω από μία κοινή διεπαφή (parser interface).
Τα βασικά χαρακτηριστικά του Apache Tika περιλαμβάνουν:

- Αυτόματη αναγνώριση MIME τύπων αρχείων.

- Εξαγωγή περιεχομένου και μεταδεδομένων από αρχείων τύπου PDF, DOCX, EML, HTML, PPT, ZIP, TXT κ.ά.
- Ενσωματωμένη υποστήριξη OCR**,** μέσω διασύνδεσης με τον μηχανισμό Tesseract, για την ανάλυση εγγράφων με ενσωματωμένο περιεχόμενο εικόνας.
- Διαθέσιμο RESTful API μέσω του Tika Server, που καθιστά δυνατή την εύκολη διασύνδεση με εξωτερικά συστήματα και την ενσωμάτωσή του σε αρχιτεκτονικές microservices.

Λόγω της ευελιξίας του, της δυνατότητας επεξεργασίας ετερογενών δεδομένων και της υποστήριξης OCR, το Tika αποτελεί ιδιαίτερα κατάλληλη επιλογή για συστήματα που απαιτούν μαζική, αυτοματοποιημένη και επεκτάσιμη επεξεργασία εγγράφων**.** Η ολοκληρωμένη αυτή προσέγγιση επιτρέπει τον συνδυασμό δομικής και οπτικής ανάλυσης περιεχομένου σε ενιαίο μηχανισμό.

3.2.3.1 Tesseract OCR (ως Υποσύστημα του Apache Tika)

Το **Tesseract OCR** αποτελεί μία από τις πλέον καταξιωμένες μηχανές **οπτικής αναγνώρισης χαρακτήρων (OCR)** ανοιχτού κώδικα, με ιστορική εξέλιξη που ξεκινά από την Hewlett-Packard και συνεχίζεται με την υποστήριξη της Google. Το σύστημα βασίζεται σε έναν πολυσταδιακό μηχανισμό αναγνώρισης, ο οποίος περιλαμβάνει φάσεις όπως **προεπεξεργασία εικόνας** (binarization, αφαίρεση θορύβου), **εντοπισμό περιοχών κειμένου και χαρακτήρων**, καθώς και **αναγνώριση χαρακτήρων μέσω LSTM (Long Short-Term Memory) νευρωνικών δικτύων** [59]. Το Tesseract υποστηρίζει περισσότερες από 100 γλώσσες, περιλαμβανομένων των ελληνικών, και παρέχει τη δυνατότητα χρήσης συνδυαστικών γλωσσικών μοντέλων (π.χ. ell+eng) για πολυγλωσσικά περιεχόμενα.

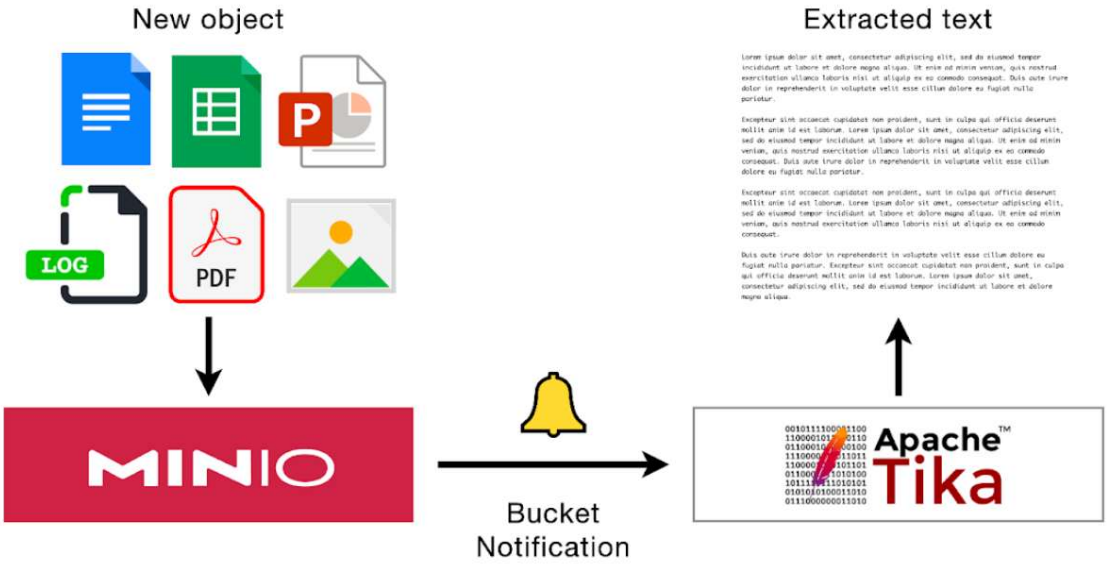
Στην παρούσα εφαρμογή, η ενσωμάτωση του Tesseract πραγματοποιείται **μέσω του Apache Tika**, το οποίο περιλαμβάνει εγγενή υποστήριξη OCR μέσω του TesseractOCRParser. Η λειτουργικότητα αυτή ενεργοποιείται **δυναμικά και αυτόματα** κατά την ανάλυση εγγράφων που δεν περιέχουν ενσωματωμένο επεξεργάσιμο κείμενο (π.χ. σαρωμένα PDF ή εικόνες τύπου JPEG/PNG). Το Tika, αφού αναγνωρίσει τον τύπο του αρχείου, επιχειρεί εξαγωγή περιεχομένου μέσω των ενσωματωμένων parsers· σε περίπτωση αποτυχίας, επιστρατεύεται ο μηχανισμός OCR χωρίς την ανάγκη εξωτερικής προγραμματιστικής παρέμβασης. Η ενσωμάτωση αυτή προσφέρει σημαντικά πλεονεκτήματα, όπως **απλοποίηση της ροής επεξεργασίας, αποφόρτιση του βασικού backend**, και **συνοχή στην ανάλυση εγγράφων ανεξαρτήτως μορφής**.

Είναι σημαντικό να σημειωθεί ότι, στο πλαίσιο της παρούσας υλοποίησης, το Tesseract χρησιμοποιείται **στην προεπιλεγμένη του διαμόρφωση**, χωρίς να πραγματοποιηθεί **εκπαίδευση νέων μοντέλων (custom training)** ή τροποποίηση των προκαθορισμένων παραμέτρων. Η απόφαση αυτή ελήφθη με γνώμονα τη διατήρηση της αρχιτεκτονικής απλότητας, καθώς τα υπάρχοντα γλωσσικά μοντέλα καλύπτουν επαρκώς τις ανάγκες της εφαρμογής για βασική αναγνώριση ελληνικών και αγγλικών εγγράφων. Παρ' όλα αυτά, η αρχιτεκτονική του συστήματος έχει σχεδιαστεί με τρόπο που επιτρέπει την **εύκολη μελλοντική επέκταση**, εφόσον κριθεί απαραίτητη η χρήση ειδικών λεξικών, domain-specific datasets ή fine-tuning.

Συνολικά, η συνεργασία του Apache Tika με το Tesseract προσφέρει μία **ενοποιημένη και αυτόνομη λύση OCR**, η οποία καθιστά δυνατή την εξαγωγή πληροφορίας ακόμη και από αρχεία χαμηλής επεξεργασιμότητας, χωρίς την ανάγκη εξειδικευμένου χειρισμού από τον χρήστη ή το backend. Το σύστημα διατηρεί υψηλό επίπεδο αυτοματοποίησης, αξιοπιστίας και ευελιξίας, συμβάλλοντας καθοριστικά στην ολοκληρωμένη και ασύγχρονη επεξεργασία αρχείων αδόμητης μορφής.

3.2.4 Συνεργατική Λειτουργία Apache Tika και MinIO

Ένας από τους κύριους λόγους για την επιλογή του **Apache Tika** στην παρούσα υλοποίηση είναι η δυνατότητά του να ενσωματώνεται απρόσκοπτα με συστήματα **αντικειμενοστραφούς αποθήκευσης**, όπως το **MinIO**, μέσω του υποσυστήματος **Tika Pipes**. Η προσέγγιση αυτή επιτρέπει την υλοποίηση ασύγχρονων αρχιτεκτονικών μεγάλης κλίμακας, στις οποίες κάθε αντικείμενο που αποθηκεύεται σε bucket μπορεί να ανακτηθεί και να υποβληθεί σε ανάλυση με ελάχιστη παρέμβαση από το κύριο backend. Ο μηχανισμός FetchEmitTuple καθορίζει πλήρως παραμετροποιήσιμες ροές: το αρχείο ανακτάται μέσω fetcher (π.χ. S3/MinIO), αποστέλλεται στον Tika Server για εξαγωγή κειμένου και μεταδεδομένων και αποθηκεύεται εκ νέου μέσω emitter σε καθορισμένη διαδρομή εξόδου. Η όλη διαδικασία υποστηρίζεται από το **/async endpoint**, το οποίο ενεργοποιεί παράλληλες εργασίες parsing. Η παρακάτω εικόνα απεικονίζει τη διαδικασία αυτοματοποιημένης επεξεργασίας αρχείων μέσω της συνεργασίας του **MinIO** με τον μηχανισμό ανάλυσης περιεχομένου **Apache Tika**.



Σχήμα 3.5: Event driven αρχιτεκτονική με MINIO και TIKA

Πηγή: MinIO and Apache Tika: A Pattern for Text Extraction (https://blog.min.io/minio-tika-text-extraction/)

Η εν λόγω αρχιτεκτονική παρουσιάζει σημαντικά πλεονεκτήματα σε περιβάλλοντα υψηλού φόρτου, καθώς επιτρέπει την **αποσύνδεση της διαδικασίας ανάλυσης από τη βασική ροή της εφαρμογής**, εξασφαλίζοντας έτσι την επεκτασιμότητα, την αξιοπιστία και την αποδοτική χρήση των υπολογιστικών πόρων. Η χρήση object storage ως ενιαίου σημείου αποθήκευσης και διανομής εγγράφων προσφέρει επιπλέον **τεχνολογική ανεξαρτησία και ευελιξία διαχείρισης**, καθώς επιτρέπει την ενοποιημένη επεξεργασία δεδομένων ανεξαρτήτως προέλευσης ή τύπου αρχείου. Η συγκεκριμένη αρχιτεκτονική έχει ήδη εφαρμοστεί σε παραγωγικά περιβάλλοντα που διαχειρίζονται εκατομμύρια έγγραφα, επιβεβαιώνοντας την καταλληλότητά της για απαιτητικά συστήματα μαζικής εξαγωγής πληροφορίας [1].

3.3 Συμπληρωματικές/υποστηρικτικές Τεχνολογίες – Τεκμηρίωση της επιλογής τους

Πέρα από τον βασικό κορμό τεχνολογιών που συγκροτούν τη λογική και αποθηκευτική υποδομή της παρούσας εφαρμογής, κρίθηκε απαραίτητη η ενσωμάτωση επιπλέον εργαλείων για την υποστήριξη ασύγχρονων διεργασιών, την καλύτερη διαχείριση πόρων και τη διατήρηση της ανταπόκρισης του συστήματος υπό συνθήκες υψηλού φόρτου. Προς αυτή την κατεύθυνση, επιλέχθηκαν οι τεχνολογίες **Redis** και **Celery**, οι οποίες συνεργάζονται ώστε να διαχειρίζονται αποτελεσματικά την ανάθεση, εκτέλεση και παρακολούθηση χρονικά απαιτητικών ή υπολογιστικά εντατικών εργασιών. Η χρήση τους δεν μεταβάλλει τον πυρήνα της εφαρμογής, αλλά ενισχύει τη λειτουργική της επεκτασιμότητα και τη σταθερότητα της εμπειρίας του τελικού χρήστη, καθιστώντας εφικτή την οργάνωση των διαδικασιών σε ουρές εργασιών και την παράλληλη επεξεργασία τους με ελεγχόμενο τρόπο.

3.3.1 Redis

Το **Redis (Remote Dictionary Server)** αποτελεί έναν από τους πιο ευρέως χρησιμοποιούμενους in-memory data stores ανοιχτού κώδικα, σχεδιασμένο για **χαμηλή καθυστέρηση**, **υψηλή απόδοση** και υποστήριξη πλούσιων δομών δεδομένων, όπως strings, lists, sets και hashes [35]. Στην παρούσα υλοποίηση, το Redis χρησιμοποιείται ως **message broker** για την υποστήριξη ασύγχρονων διεργασιών μέσω του πλαισίου **Celery**, διαμορφώνοντας ένα σταθερό υπόβαθρο για task queuing, παρακολούθηση και διαχείριση κατανεμημένων εργασιών [36].

Σχήμα 3.6: Λογότυπο του Redis

Πηγή: Official Redis Website (https://brand.redis.io/d/wUTSgvB5PCyH/how-we-look#/visual-guidelines/logo)

Ωστόσο, η λειτουργία του Redis στην αρχιτεκτονική της εφαρμογής δεν περιορίζεται μόνο στην ανάθεση εργασιών. Αντιθέτως, αξιοποιείται επίσης για τις εξής κρίσιμες λειτουργίες:

1. **Caching ενδιάμεσων αποτελεσμάτων**, ιδίως σε υπολογιστικά εντατικές διεργασίες όπως σύνθετα ερωτήματα βάσης δεδομένων (DB queries), προσφέροντας **χαμηλή καθυστέρηση απόκρισης (low latency)** και **αποσυμφόρηση του backend**, ιδίως σε περιπτώσεις επαναλαμβανόμενης πρόσβασης σε ίδια δεδομένα. [37].
2. **Αποθήκευση και διάδοση συμβάντων (event handling)**: Το Redis λειτουργεί ως buffer για τα **object event notifications** που εκπέμπονται από το σύστημα αποθήκευσης MinIO, καταγράφοντας ενέργειες όπως νέες μεταφορτώσεις αρχείων ή τροποποιήσεις, οι οποίες ενεργοποιούν διαδοχικές ενέργειες επεξεργασίας (π.χ. ενεργοποίηση του Tika) [38].

3. **Προσωρινή διατήρηση κατάστασης (state tracking)**: Χρησιμοποιείται για την αποθήκευση προσωρινών μεταβλητών κατά την εκτέλεση εργασιών, επιτρέποντας λειτουργίες όπως progress tracking, estimation of completion time και retry mechanisms [39].
4. **Διαμεσολάβηση σε WebSocket επικοινωνία**, μέσω των pub/sub δυνατοτήτων του, επιτρέποντας την **αξιόπιστη και σε πραγματικό χρόνο μετάδοση μηνυμάτων** σε περιβάλλοντα με πολλαπλά backend instances, όπως σε εφαρμογές με **Django Channels** και real-time λειτουργικότητα.

Η επιλογή του Redis ενισχύθηκε περαιτέρω από τις εγγενείς δυνατότητές του για **persistence (μέσω RDB ή AOF)**, τους **Pub/Sub μηχανισμούς επικοινωνίας** που διευκολύνουν real-time ενημερώσεις, καθώς και τη δυνατότητα αποθήκευσης δεδομένων με χρονικά όρια (TTL), εξαιρετικά χρήσιμη για προσωρινές ροές πληροφορίας [40].

Συνολικά, το Redis αποτελεί θεμέλιο για την αρχιτεκτονική **χαλαρής σύζευξης (loose coupling)** της εφαρμογής, επιτρέποντας αποσύνδεση των υποσυστημάτων, ασύγχρονη επεξεργασία και επεκτασιμότητα σε πραγματικό χρόνο.

3.3.2 Celery

Το **Celery** (Distributed Task Queue for Python) αποτελεί ένα ισχυρό και ευρέως χρησιμοποιούμενο εργαλείο για την υλοποίηση **ασύγχρονων και κατανεμημένων διεργασιών** (asynchronous distributed tasks) σε εφαρμογές Python. Ειδικά σε περιβάλλοντα που βασίζονται στο Django, το Celery προσφέρει **άψογη ενσωμάτωση**, καθιστώντας δυνατή την εκτέλεση χρονοβόρων διεργασιών στο παρασκήνιο, χωρίς να επιβαρύνεται η εμπειρία χρήστη ή να καθυστερεί η αποκρισιμότητα του συστήματος [31].

Σχήμα 3.7: Λογότυπο του Celery

Πηγή: Official Celery Website (https://docs.celeryq.dev/en/stable/index.html)

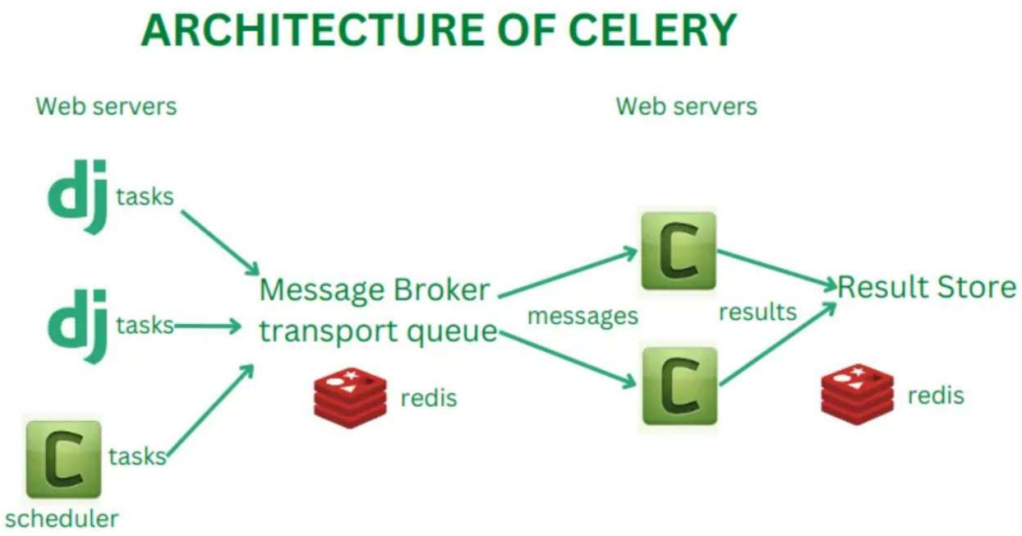
Το Celery βασίζεται στο μοντέλο **παραγωγού–καταναλωτή (producer–consumer)**. Το backend της εφαρμογής (Django) αναλαμβάνει τον ρόλο του παραγωγού (producer), καταχωρώντας τις εργασίες σε ένα message broker (Redis), ενώ οι **εργάτες (Celery workers)** λειτουργούν ως καταναλωτές (consumers), ανακτώντας και εκτελώντας τις εργασίες με ανεξαρτησία και παραλληλία. Οι εργασίες μπορεί να αφορούν λειτουργίες αυξημένου υπολογιστικού κόστους όπως εννοιολογικό εμπλουτισμό (NER), scheduled indexing ή άλλες επεξεργασίες φυσικής γλώσσας.

Βασικά πλεονεκτήματα της χρήσης Celery περιλαμβάνουν:

1. **Ασύγχρονη εκτέλεση**, η οποία αποφορτίζει τον web server από χρονοβόρες διεργασίες.

2. **Υποστήριξη πολλαπλών workers**, που επιτρέπει τη **μαζική και παράλληλη επεξεργασία** μεγάλου αριθμού αιτημάτων.
3. **Συμβατότητα με Redis**, τόσο ως message broker όσο και ως προσωρινός χώρος αποθήκευσης (result store).
4. **Δυνατότητα προγραμματισμένων εργασιών (periodic/scheduled tasks)**, μέσω εργαλείων όπως το Celery Beat.
5. **Διαχείριση σύνθετων workflows** με υποστήριξη για retry policies, time limits, task chaining κ.ά. [32].

Η παρακάτω αρχιτεκτονική απεικόνιση συνοψίζει την ενσωμάτωση του Celery στην εφαρμογή:



Σχήμα 3.8: Αρχιτεκτονική του Celery με Redis

Πηγή: Asynchronous Task Execution with Celery in Django

(https://medium.com/@veronicakylie1/asynchronous-task-execution-with-celery-in-django-e54aa6e1f34f)

1. **Django**: Δέχεται το αίτημα του χρήστη και δημιουργεί ένα task.
2. **Redis (Broker)**: Παρέχει τον μηχανισμό μεταφοράς και αποθήκευσης των tasks.
3. **Celery Workers**: Εκτελούν τις διεργασίες, επιστρέφοντας αποτελέσματα στο backend.
4. **Redis (Result Store)**: Αποθηκεύει προσωρινά τα αποτελέσματα των εργασιών.
5. **Scheduler**: Διαχειρίζεται επαναλαμβανόμενα tasks, π.χ. περιοδικό έλεγχο αρχείων [32].

Η ενσωμάτωση του Celery ενισχύει καθοριστικά την **αξιοπιστία, παραλληλία και επεκτασιμότητα** της εφαρμογής, καθιστώντας δυνατή την υποστήριξη φορτισμένων σεναρίων χρήσης και σύνθετων υπολογιστικών ροών. Παράλληλα, η τεχνολογία Celery θεωρείται **best practice** για Python/Django εφαρμογές με υψηλές απαιτήσεις απόδοσης και ασύγχρονης λειτουργίας [33], [34].

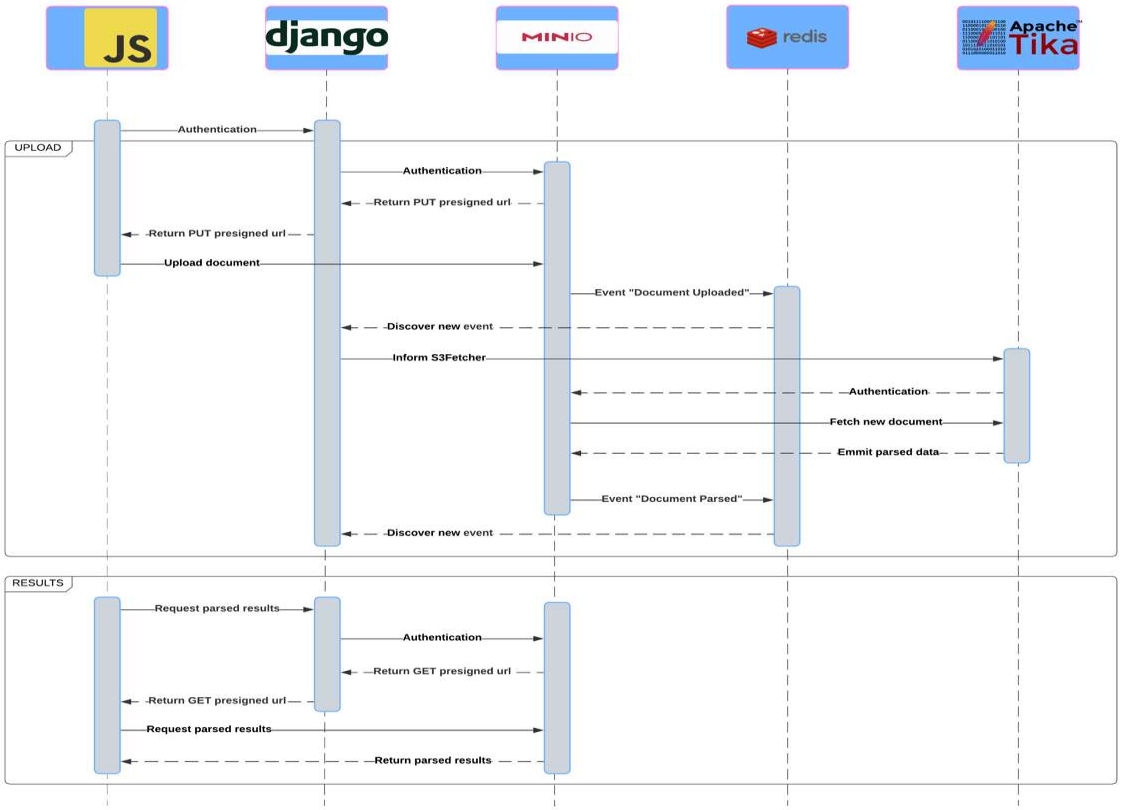
3.4 Διάγραμμα Ακολουθιών (Sequence Diagram) – Ροή Document Upload / Data extraction

Το παρακάτω διάγραμμα ακολουθιών (sequence diagram) απεικονίζει τη ροή δεδομένων και την αλληλεπίδραση μεταξύ των βασικών υποσυστημάτων της διαδικτυακής εφαρμογής κατά τη διαδικασία

μεταφόρτωσης εγγράφων, εξαγωγής περιεχομένου και ανάκτησης των επεξεργασμένων αποτελεσμάτων από τον τελικό χρήστη.

Η διαδικασία εκκινείται με την αυθεντικοποίηση του χρήστη και ολοκληρώνεται με την ασφαλή προβολή των δομημένων δεδομένων, όπως έχουν εξαχθεί από το Apache Tika και αποθηκευτεί στο MinIO. Αναλυτικά, τα στάδια της ροής έχουν ως εξής:

1. **Αυθεντικοποίηση Χρήστη**
   Ο client (frontend σε JavaScript) αποστέλλει αίτημα προς το backend (Django) για τη μεταφόρτωση εγγράφου. Πραγματοποιείται έλεγχος ταυτότητας του χρήστη βάσει των μηχανισμών αυθεντικοποίησης του συστήματος.
2. **Δημιουργία Presigned URL (PUT)**
   Το Django backend δημιουργεί έναν presigned URL μέσω της βιβλιοθήκης MinIO SDK ή Boto3, παρέχοντας στον client τη δυνατότητα απευθείας μεταφόρτωσης του αρχείου στο MinIO, χωρίς την παρεμβολή του backend κατά τη μεταφορά.
3. **Μεταφόρτωση Εγγράφου**
   Ο χρήστης ανεβάζει το έγγραφο στο σύστημα αποθήκευσης (MinIO), αξιοποιώντας τον presigned URL που δημιουργήθηκε στο προηγούμενο βήμα.
4. **Ανίχνευση Συμβάντος "Document Uploaded"**
   Η επιτυχής αποθήκευση του αρχείου πυροδοτεί την ενεργοποίηση event τύπου *Document Uploaded* στο MinIO. Το event καταγράφεται στο Redis μέσω του συστήματος ειδοποιήσεων (bucket notification hooks).
5. **Υποβολή Εργασίας στο Apache Tika**
   Ένας Celery worker (task consumer) παρακολουθεί το Redis για νέα συμβάντα. Μόλις εντοπιστεί νέο *Document Uploaded* event, ο worker συντάσσει και υποβάλλει αίτημα προς το endpoint /async του Apache Tika, βασισμένο στη δομή *FetchEmitTuple*.
6. **Επεξεργασία και Αποθήκευση Εξαγόμενων Δεδομένων**
   Το Tika, μέσω του *S3Fetcher*, ανακτά το αρχείο από το MinIO και εκτελεί τις διαδικασίες ανάλυσης περιεχομένου και εξαγωγής μεταδεδομένων (με χρήση OCR εφόσον απαιτείται). Τα εξαγόμενα δεδομένα αποθηκεύονται από τον *Emitter* ως αρχείο .json σε διαφορετικό bucket (parsed-documents), με πλήρη αναφορά στο αρχικό αρχείο.
7. **Καταγραφή Συμβάντος "Document Parsed"**
   Μετά την επιτυχή αποθήκευση του .json αρχείου, δημιουργείται νέο event τύπου *Document Parsed*, το οποίο αποθηκεύεται στο Redis. Το συμβάν αυτό καθιστά δυνατή την ειδοποίηση του backend για την ολοκλήρωση της επεξεργασίας.
8. **Ανάκτηση Επεξεργασμένων Δεδομένων από τον Χρήστη**
   Όταν ο χρήστης ζητήσει προβολή αποτελεσμάτων, το backend δημιουργεί έναν presigned URL (GET) για το αντίστοιχο .json αρχείο στο bucket parsed-documents και το αποστέλλει στον client. Η προβολή γίνεται μέσω της διεπαφής χρήστη, σε κατάλληλα δομημένη μορφή.



Σχήμα 3.9: Διάγραμμα ακολουθιών Document Upload / Data extraction

Η παρακάτω ακολουθία παρουσιάζει τα βήματα που εκτελούνται κατά τη μεταφόρτωση εγγράφου, την αυτόματη επεξεργασία του μέσω Apache Tika και την ανάκτηση των αποτελεσμάτων από τον τελικό χρήστη. Τα βέλη δηλώνουν την κατεύθυνση της επικοινωνίας μεταξύ των βασικών υποσυστημάτων της εφαρμογής:

[JS Frontend] → [Django Backend]
Ο χρήστης αυθεντικοποιείται και αποστέλλει αίτημα για μεταφόρτωση εγγράφου.
[Django Backend] → [MinIO]
Το backend δημιουργεί έναν *presigned PUT URL* μέσω MinIO SDK και τον επιστρέφει στον client.
[JS Frontend] → [MinIO]
Ο client χρησιμοποιεί τον presigned URL για να ανεβάσει το έγγραφο απευθείας στο MinIO.
[MinIO] → [Redis]
Η επιτυχής μεταφόρτωση πυροδοτεί ένα event τύπου *Document Uploaded*, το οποίο αποστέλλεται στο Redis μέσω συστήματος ειδοποιήσεων (bucket notifications).
[Redis] → [Celery Worker - S3Fetcher]
Ένας worker του Django/Celery εντοπίζει το συμβάν και ενεργοποιεί τη ροή επεξεργασίας.
[S3Fetcher] → [MinIO] (Authentication)
Ο worker αυθεντικοποιείται στο MinIO για να αποκτήσει πρόσβαση στο αποθηκευμένο αρχείο.
[S3Fetcher] → [MinIO] (Fetch)
Το αρχείο ανακτάται από το MinIO μέσω του S3Fetcher.
[S3Fetcher] → [Apache Tika]

Υποβάλλεται αίτημα προς το /async endpoint του Tika Server, το οποίο αναλαμβάνει την ανάλυση του αρχείου (με ή χωρίς OCR).

[Apache Tika] → [MinIO] (Emit)

Το εξαγόμενο περιεχόμενο και τα μεταδεδομένα αποθηκεύονται ως .json αρχείο σε διακριτό bucket (π.χ. parsed-documents) μέσω του Tika Emitter.

[Apache Tika] → [Redis]

Με την επιτυχή αποθήκευση, καταγράφεται νέο event τύπου *Document Parsed* στο Redis.

[JS Frontend] → [Django Backend]

Ο χρήστης ζητά την προβολή των αποτελεσμάτων για το επεξεργασμένο αρχείο.

[Django Backend] → [MinIO] (GET)

Το backend δημιουργεί έναν *presigned GET URL* για την ασφαλή ανάκτηση του αρχείου .json με τα επεξεργασμένα δεδομένα.

[JS Frontend] → [MinIO]

Ο client κατεβάζει το αρχείο αποτελεσμάτων από το bucket parsed-documents.

[JS Frontend] → [UI]

Τα αποτελέσματα εμφανίζονται στο χρήστη μέσω της διεπαφής (UI), σε δομημένη μορφή.

3.5 Επίλογος

Το πρώτο στάδιο ανάπτυξης υλοποίησε έναν πλήρως αυτοματοποιημένο και επεκτάσιμο μηχανισμό εισαγωγής και επεξεργασίας εγγράφων, ικανό να διαχειρίζεται ετερογενείς μορφότυπους και να εξάγει χρήσιμη πληροφορία με συνέπεια και ακρίβεια. Η σύμπραξη τεχνολογιών όπως το Django, το MinIO, το Apache Tika, υποστηριζόμενων από Redis και Celery για ασύγχρονη εκτέλεση, διαμόρφωσε μια αρχιτεκτονική που ανταποκρίνεται σε υψηλές απαιτήσεις απόδοσης, ασφάλειας και κλιμάκωσης.

Η υλοποίηση της ροής μεταφόρτωσης και εξαγωγής δεδομένων, βασισμένη σε event-driven σχεδιασμό και presigned URL μεταφορές, διασφαλίζει την ελάχιστη επιβάρυνση του backend και την άμεση διαθεσιμότητα των αποτελεσμάτων στον χρήστη. Η προσέγγιση αυτή παρέχει θεμέλιο για την περαιτέρω ενσωμάτωση προηγμένων λειτουργιών, όπως η ανάλυση περιεχομένου και η σημασιολογική αναζήτηση.

Στο επόμενο κεφάλαιο, η ανάλυση εστιάζει στη δεύτερη φάση ανάπτυξης, όπου τα επεξεργασμένα δεδομένα αξιοποιούνται για την υλοποίηση μηχανισμών αναζήτησης και ανακάλυψης πληροφορίας, επεκτείνοντας τη λειτουργικότητα της εφαρμογής πέρα από την απλή αποθήκευση και προβολή περιεχομένου.

Κεφάλαιο 4ο: Δεύτερο στάδιο ανάπτυξης – Αποθήκευση, Ανάκτηση και Εξαγωγή Πληροφορίας (Data Storage / Retrieval & Information Extraction)

4.1 Εισαγωγή

Το δεύτερο λειτουργικό στάδιο της εφαρμογής επικεντρώνεται στη διαχείριση, αποθήκευση και σημασιολογική επεξεργασία των δεδομένων που έχουν ήδη εξαχθεί στο πρώτο στάδιο. Εδώ, το σύστημα μεταβαίνει από την απλή εξαγωγή περιεχομένου στην οργάνωση και εμπλουτισμό της πληροφορίας, με στόχο την αποδοτική αναζήτηση, την αναγνώριση οντοτήτων και τη δημιουργία γραφημάτων γνώσης.
Η υλοποίηση αξιοποιεί πολλαπλά υποσυστήματα αποθήκευσης και αναζήτησης – όπως PostgreSQL, OpenSearch, Qdrant και Neo4j – καθώς και μηχανισμούς εξαγωγής πληροφορίας όπως το spaCy για Named Entity Recognition (NER) και το Ollama για παραγωγή embeddings.
Η ροή των διεργασιών σε αυτό το στάδιο δεν είναι αυστηρά σειριακή αλλά ασύγχρονα ενορχηστρωμένη, επιτρέποντας σε διαφορετικά βήματα να εκτελούνται ταυτόχρονα και να ολοκληρώνονται ανεξάρτητα. Μέσω του γραφικού περιβάλλοντος, ο χρήστης έχει συνεχή εικόνα της προόδου και μπορεί να παρεμβαίνει για επιβεβαίωση ή διόρθωση δεδομένων, βελτιώνοντας έτσι την ακρίβεια και την ποιότητα των αποτελεσμάτων.

4.2 Βασικές Τεχνολογίες Αποθήκευσης και Ανάκτησης Δεδομένων

Η αρχιτεκτονική της εφαρμογής στηρίζεται σε ένα συνδυασμό τεχνολογιών αποθήκευσης και ανάκτησης δεδομένων, οι οποίες καλύπτουν διαφορετικές λειτουργικές ανάγκες και συνεργάζονται για να προσφέρουν υψηλή απόδοση, ακρίβεια και ευελιξία. Στο πλαίσιο αυτό, επιλέχθηκαν και ενσωματώθηκαν η **Elasticsearch/OpenSearch**, η **PostgreSQL**, η **Qdrant** και η **Neo4j**.
Η κάθε τεχνολογία εξυπηρετεί διαφορετικό ρόλο:

- Το **OpenSearch** λειτουργεί ως μηχανή αναζήτησης και ευρετηρίασης για λέξεις-κλειδιά και δομημένα queries.
- Η **PostgreSQL** αποτελεί το κεντρικό αποθετήριο δομημένων δεδομένων, διασφαλίζοντας ακεραιότητα και αξιοπιστία.
- Η **Qdrant** εξυπηρετεί την αναζήτηση βασισμένη σε σημασιολογική ομοιότητα, αξιοποιώντας διανυσματικές αναπαραστάσεις (embeddings).
- Η **Neo4j** υποστηρίζει την αναπαράσταση και ανάλυση συσχετίσεων μέσω γράφων.

Η συνέργεια αυτών των τεχνολογιών επιτρέπει στο σύστημα να καλύπτει όλο το φάσμα αναζητήσεων: από την κλασική αναζήτηση βάσει λέξεων-κλειδιών, μέχρι την σημασιολογική ανάκτηση πληροφορίας και την εξερεύνηση περίπλοκων δικτύων δεδομένων.

4.2.1 Elasticsearch και OpenSearch

Η **Elasticsearch** επιλέχθηκε ως το κεντρικό υποσύστημα της εφαρμογής, με σκοπό την αποδοτική αναζήτηση σε μεγάλα και ετερογενή σύνολα δεδομένων, δίνοντας έμφαση στην ταχύτητα απόκρισης, την πληρότητα των αποτελεσμάτων και την υποστήριξη προχωρημένων τεχνικών αναζήτησης, όπως το *fuzzy matching*.

Σχήμα 4.1: Λογότυπο του Opensearch

Πηγή: Official Opensearch Website (*https://opensearch.org/trademark-brand-policy/)*

Η **Elasticsearch** επιλέχθηκε ως το κεντρικό υποσύστημα της εφαρμογής, με σκοπό την αποδοτική αναζήτηση σε μεγάλα και ετερογενή σύνολα δεδομένων, δίνοντας έμφαση στην ταχύτητα απόκρισης, την πληρότητα των αποτελεσμάτων και την υποστήριξη προχωρημένων τεχνικών αναζήτησης, όπως το *fuzzy matching*.
Στο πλαίσιο της υλοποίησης, λειτουργεί ως ενιαία μηχανή αναζήτησης πάνω σε όλο το πληροφοριακό απόθεμα της εφαρμογής, παρέχοντας παράλληλα δυνατότητες αυτόματης συμπλήρωσης (autocomplete) και φιλτραρίσματος βάσει μεταδεδομένων. [28].
Τα κύρια χαρακτηριστικά που καθιστούν την Elasticsearch κατάλληλη για την παρούσα εφαρμογή περιλαμβάνουν:

- Κατανεμημένη αρχιτεκτονική, η οποία υποστηρίζει την κλιμάκωση σε μεγάλους όγκους δεδομένων, εξασφαλίζοντας αξιοπιστία και υψηλή διαθεσιμότητα.
- Δυνατότητα *real-time* indexing και αναζήτησης, επιτρέποντας την άμεση εμφάνιση νέων ή ενημερωμένων δεδομένων στα αποτελέσματα.
- Υποστήριξη για *fuzzy queries*, *range filters* και *text analysis*, διευκολύνοντας την αναζήτηση ακόμα και σε περιπτώσεις ανορθογραφιών ή ελλιπών όρων.
- Ενσωμάτωση με το Django μέσω εξειδικευμένων βιβλιοθηκών (π.χ. *django-elasticsearch-dsl*) που αυτοματοποιούν τη δημιουργία και τον συγχρονισμό των index.

Ιδιαίτερα σημαντική είναι η λειτουργία της αυτόματης συμπλήρωσης, η οποία υλοποιείται μέσω ειδικών τεχνικών ανάλυσης κειμένου, όπως οι *edge n-gram analyzers*, που επιτρέπουν την πρόβλεψη πιθανών αποτελεσμάτων με βάση τα πρώτα γράμματα ή λέξεις που εισάγει ο χρήστης [29]. Αυτό ενισχύει σημαντικά τη χρηστικότητα του συστήματος, επιτρέποντας τον εντοπισμό εγγράφων, οντοτήτων ή υποθέσεων ακόμη και με ελάχιστα δεδομένα εισόδου.
Στην παρούσα εφαρμογή, η λειτουργικότητα αυτή υλοποιείται μέσω του **OpenSearch**, ενός πλήρως συμβατού *fork* της Elasticsearch, το οποίο δημιουργήθηκε και συντηρείται από την κοινότητα, προσφέροντας ανοιχτό κώδικα και διαρκείς βελτιώσεις. Το OpenSearch, πέρα από τη διατήρηση της πλήρους συμβατότητας με τα API και τις λειτουργίες της Elasticsearch, παρέχει προηγμένες δυνατότητες ασφάλειας, βελτιωμένες επιδόσεις στην αναζήτηση, καθώς και ενσωματωμένα εργαλεία

για οπτικοποίηση δεδομένων. Η τεχνολογία αυτή προσφέρει βελτιστοποιημένη απόδοση κατά την αναζήτηση εγγράφων, υποθέσεων και οντοτήτων, επιτρέποντας έξυπνη ανάκτηση πληροφορίας σε πραγματικό χρόνο [30].

4.2.1.1 Συγκριτική αξιολόγηση OpenSearch και Elasticsearch

Η επιλογή του OpenSearch έναντι των πρόσφατων εκδόσεων της Elasticsearch βασίστηκε σε τεχνικές, λειτουργικές και νομικές παραμέτρους. Από το 2021, η Elasticsearch, υπό την ανάπτυξη της Elastic NV, άλλαξε την άδειά της από την *Apache 2.0* σε συνδυασμό των *Server Side Public License* (SSPL) και *Elastic License*, οι οποίες επιβάλλουν περιορισμούς στη χρήση και την αναδιανομή. Αντιθέτως, το OpenSearch, ως *fork* που προήλθε από την έκδοση 7.10 της Elasticsearch, συνεχίζει να διανέμεται υπό την άδεια *Apache 2.0*, διασφαλίζοντας πλήρη ελευθερία χρήσης, τροποποίησης και ενσωμάτωσης σε εμπορικά και ερευνητικά έργα χωρίς νομικούς περιορισμούς [28].

Σε τεχνικό επίπεδο, το OpenSearch διατηρεί πλήρη συμβατότητα με τα API της Elasticsearch έως την έκδοση 7.10, επιτρέποντας την απευθείας μεταφορά υφιστάμενων εφαρμογών χωρίς εκτεταμένες αλλαγές κώδικα. Παράλληλα, παρέχει ενσωματωμένες δυνατότητες ασφαλείας, όπως έλεγχο πρόσβασης βάσει ρόλων (*Role-Based Access Control – RBAC*), κρυπτογράφηση επικοινωνίας και υποστήριξη εξωτερικών μηχανισμών ταυτοποίησης (LDAP, SAML, OpenID Connect).

Λειτουργικά, η κοινότητα ανάπτυξης του OpenSearch εστιάζει στη διαφάνεια, την τακτική έκδοση ενημερώσεων και την ενίσχυση χαρακτηριστικών που σχετίζονται με την αναζήτηση σε μεγάλα και κατανεμημένα σύνολα δεδομένων. Επίσης, ενσωματώνει τα *OpenSearch Dashboards*, ένα εργαλείο οπτικοποίησης ισοδύναμο με το Kibana, προσφέροντας δυνατότητες παρακολούθησης και ανάλυσης δεδομένων σε πραγματικό χρόνο [30].

Συνεπώς, η επιλογή του OpenSearch προσφέρει στην εφαρμογή τα πλεονεκτήματα της πλήρους ανοικτότητας, της συμβατότητας με υπάρχοντα πρότυπα ανάπτυξης και της απουσίας νομικών περιορισμών, ενώ ταυτόχρονα διατηρεί υψηλές επιδόσεις και τις προηγμένες δυνατότητες που απαιτούνται για ένα σύστημα αναζήτησης εγγράφων, οντοτήτων και συσχετίσεων μεγάλης κλίμακας [29].

4.2.2 PostgreSQL

Η **PostgreSQL** επιλέχθηκε ως το κύριο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) της εφαρμογής, λόγω της σταθερότητας, της επεκτασιμότητας και της πλήρους υποστήριξης προηγμένων λειτουργιών που απαιτούνται σε σύνθετα πληροφοριακά συστήματα. Πρόκειται για ένα ώριμο έργο ανοικτού κώδικα, με συνεχή ανάπτυξη από ενεργή κοινότητα και πολυετή παρουσία στον χώρο των enterprise εφαρμογών, το οποίο υποστηρίζει το πρότυπο SQL και παρέχει εκτεταμένες δυνατότητες για αποθήκευση, διαχείριση και ανάλυση δεδομένων [31].

Σχήμα 4.2: Λογότυπο της PostgreSQL

Πηγή: Official PostgreSQL Website (*https://www.postgresql.org/about/press/presskit16/base/)*

Η απόφαση για την υιοθέτηση της PostgreSQL βασίστηκε σε συγκεκριμένα πλεονεκτήματα:

- **Αξιοπιστία και Σταθερότητα**: Διαθέτει μηχανισμούς *ACID compliance*, προηγμένη διαχείριση συναλλαγών και ισχυρή υποστήριξη για point-in-time recovery, εξασφαλίζοντας ακεραιότητα δεδομένων ακόμη και σε περιπτώσεις αστοχίας συστήματος.
- **Πλούσια Υποστήριξη Τύπων Δεδομένων**: Εκτός από τους τυπικούς τύπους SQL, υποστηρίζει JSON/JSONB, XML, γεωχωρικά δεδομένα (μέσω του *PostGIS*), καθώς και χρήστη-ορισμένους τύπους δεδομένων.
- **Επεκτασιμότητα**: Δίνει τη δυνατότητα δημιουργίας και χρήσης επεκτάσεων (*extensions*) και αποθηκευμένων διαδικασιών σε πολλές γλώσσες (SQL, PL/pgSQL, Python κ.ά.).
- **Προηγμένες Δυνατότητες Αναζήτησης**: Περιλαμβάνει πλήρη υποστήριξη *full-text search*, *GIN* και *GiST* index structures για υψηλές επιδόσεις σε πολύπλοκα ερωτήματα.
- **Κλιμάκωση και Απόδοση**: Υποστηρίζει παράλληλη εκτέλεση ερωτημάτων, κατανεμημένες βάσεις δεδομένων (sharding) και replication, καλύπτοντας ανάγκες τόσο για υψηλή διαθεσιμότητα όσο και για αντοχή σε μεγάλα φορτία.
- **Ενσωμάτωση με Django**: Μέσω του *Django ORM*, η PostgreSQL αξιοποιείται πλήρως, υποστηρίζοντας σύνθετα ερωτήματα, constraints, triggers και stored procedures με φυσικό και απρόσκοπτο τρόπο [32].

Η PostgreSQL χρησιμοποιείται στην εφαρμογή για την αποθήκευση και διαχείριση του δομημένου πυρήνα δεδομένων, που περιλαμβάνει χρήστες, ρόλους, δικαιώματα πρόσβασης, υποθέσεις έγγραφα, οντότητες και τις μεταξύ τους σχέσεις. Η χρήση της επιτρέπει την εφαρμογή σύνθετων περιορισμών ακεραιότητας (constraints) και την αξιοποίηση triggers για την αυτοματοποίηση διαδικασιών, ενώ παράλληλα προσφέρει σταθερή βάση για την ενσωμάτωση με άλλα υποσυστήματα, όπως το OpenSearch για προηγμένη αναζήτηση ή το Neo4j για γραφηματική αναπαράσταση δεδομένων.

Η υιοθέτηση της PostgreSQL ενισχύει την αξιοπιστία και τη λειτουργική πληρότητα της εφαρμογής, ενώ η ανοιχτή της φύση επιτρέπει την πλήρη προσαρμογή στις εκάστοτε απαιτήσεις του έργου. Σε συνδυασμό με την ισχυρή υποστήριξη από το οικοσύστημα του Django, προσφέρει ένα ευέλικτο, ασφαλές και επεκτάσιμο περιβάλλον για την αποθήκευση και διαχείριση κρίσιμων δεδομένων [33].

4.2.3 Qdrant

Η **Qdrant** (*Quantized Dense RetrievAl eNgine*) επιλέχθηκε ως η βασική μηχανή διανυσματικής αναζήτησης της εφαρμογής, παρέχοντας προηγμένες δυνατότητες αποθήκευσης, αναζήτησης και κατάταξης δεδομένων με βάση τη σημασιολογική εγγύτητα σε διανυσματικό χώρο [34]. Πρόκειται για μία σύγχρονη, ανοικτού κώδικα πλατφόρμα (*Apache 2.0 license*) που σχεδιάστηκε ειδικά για εφαρμογές *vector search* μεγάλης κλίμακας, καθιστώντας την ιδανική επιλογή για έργα που απαιτούν ανακτήσεις πληροφορίας βασισμένες στη σημασιολογική ομοιότητα και όχι μόνο στη σύμπτωση λέξεων-κλειδιών. [35].

Σχήμα 4.3: Λογότυπο της Qdrant

Πηγή: Official Qdrant Website (*https://qdrant.tech/brand-resources/)*

Η απόφαση για υιοθέτηση της Qdrant βασίστηκε σε συγκεκριμένα πλεονεκτήματα:

- **Υψηλή Απόδοση και Κλιμάκωση**: Βελτιστοποιημένη για αποθήκευση και αναζήτηση δισεκατομμυρίων διανυσμάτων, με υποστήριξη *Approximate Nearest Neighbor (ANN)* αναζήτησης μέσω του αλγορίθμου HNSW (*Hierarchical Navigable Small World graphs*) [36].
- **Σύνδεση Μεταδεδομένων και Διανυσμάτων**: Δυνατότητα αποθήκευσης επιπλέον πεδίων (π.χ. κατηγορία, ημερομηνία) που συνδέονται με κάθε διάνυσμα, επιτρέποντας υβριδικές αναζητήσεις (συνδυασμός κειμένου και embeddings).
- **Real-time Ενημέρωση**: Υποστήριξη δυναμικής εισαγωγής, διαγραφής και ενημέρωσης εγγραφών χωρίς ανάγκη επανευρετηρίασης.
- **Ευκολία Ενοποίησης**: Παρέχει REST API και gRPC endpoints, καθώς και επίσημες βιβλιοθήκες για Python, Go και JavaScript.
- **Σύνθετο Φιλτράρισμα**: Δυνατότητα συνδυασμού φίλτρων και αναζήτησης με ομοιότητα, ιδανικό για σενάρια όπου απαιτείται σημασιολογική αναζήτηση περιορισμένη σε συγκεκριμένα υποσύνολα δεδομένων.

Στην παρούσα εφαρμογή, η Qdrant χρησιμοποιείται για την υλοποίηση σημασιολογικής αναζήτησης πάνω σε δεδομένα που έχουν υποστεί επεξεργασία μέσω NLP pipelines (π.χ. εξαγωγή *embeddings* από πολυγλωσσικά μοντέλα). Κάθε έγγραφο μετατρέπεται σε διανυσματική αναπαράσταση, η οποία αποθηκεύεται στην Qdrant. Με αυτόν τον τρόπο, το σύστημα μπορεί να αναζητά πληροφορίες με βάση το νόημα του κειμένου, ακόμη και όταν οι όροι αναζήτησης δεν ταυτίζονται ακριβώς με αυτούς που περιέχονται στο περιεχόμενο [53].

Η Qdrant λειτουργεί συμπληρωματικά με το OpenSearch: το OpenSearch καλύπτει την αναζήτηση βάσει λέξεων-κλειδιών (*keyword search*), ενώ η Qdrant καλύπτει την αναζήτηση βάσει σημασιολογικής ομοιότητας (*semantic similarity search*), παρέχοντας μία ολοκληρωμένη εμπειρία αναζήτησης [34].

Η ενσωμάτωση της Qdrant ενισχύει σημαντικά την ικανότητα του συστήματος να χειρίζεται ερωτήματα φυσικής γλώσσας και να ανακτά πληροφορίες με μεγαλύτερη ακρίβεια και συνάφεια. Χάρη στην ανοικτή της φύση, στην ενεργή κοινότητα ανάπτυξης και στη δυνατότητα λειτουργίας τόσο σε τοπικό περιβάλλον όσο και σε περιβάλλον cloud, αποτελεί κρίσιμο πυλώνα για την υλοποίηση σύγχρονων μηχανισμών αναζήτησης βασισμένων σε τεχνολογίες Τεχνητής Νοημοσύνης [36], .

4.2.4 Neo4j

Η **Neo4j** επιλέχθηκε ως το βασικό σύστημα γραφηματικής βάσης δεδομένων (*graph database*) της εφαρμογής, με στόχο την αποθήκευση, διαχείριση και ανάλυση οντοτήτων και των μεταξύ τους σχέσεων. Αποτελεί την πλέον διαδεδομένη πλατφόρμα γραφηματικών βάσεων, με ώριμο οικοσύστημα εργαλείων, υψηλή απόδοση και μεγάλη ευελιξία, ιδανική για εφαρμογές που απαιτούν πολύπλοκη μοντελοποίηση και ανάλυση δικτύων πληροφορίας [37].

Σχήμα 4.4: Λογότυπο της Neo4j

Πηγή: Official Neo4j Website (https://neo4j.com/)

Η επιλογή της Neo4j βασίστηκε σε συγκεκριμένα πλεονεκτήματα:

- **Φυσική Μοντελοποίηση Σχέσεων**: Τα δεδομένα αναπαρίστανται ως κόμβοι (*nodes*) και ακμές (*relationships*), επιτρέποντας την απευθείας αποτύπωση πραγματικών σχέσεων χωρίς την πολυπλοκότητα των *joins* που απαιτούνται σε σχεσιακές βάσεις.
- **Υψηλή Απόδοση σε Σύνθετα Ερωτήματα**: Η Neo4j αξιοποιεί το μοντέλο *index-free adjacency*, που επιτρέπει άμεση πλοήγηση από κόμβο σε κόμβο ακόμη και σε πολύ μεγάλα γραφήματα.
- **Εκφραστική Γλώσσα Ερωτημάτων**: Η *Cypher Query Language (CQL)* προσφέρει υψηλή αναγνωσιμότητα και εκφραστικότητα για πολύπλοκα ερωτήματα.
- **Κλιμάκωση και Υψηλή Διαθεσιμότητα**: Υποστηρίζει clustering, replication και μηχανισμούς αυτόματης ανάκαμψης για περιβάλλοντα παραγωγής.
- **Ενσωμάτωση με Python και Django**: Μέσω βιβλιοθηκών όπως *py2neo* και *neo4j-driver*, καθίσταται δυνατή η απρόσκοπτη επικοινωνία με την εφαρμογή και η δυναμική ενημέρωση των δεδομένων [54], [38].

Στην παρούσα εφαρμογή, η Neo4j αξιοποιείται ως εξειδικευμένο υποσύστημα αποθήκευσης και διαχείρισης γραφημάτων γνώσης, καταγράφοντας τα αποτελέσματα των διαδικασιών αναγνώρισης οντοτήτων (Named Entity Recognition – NER) και εξαγωγής σχέσεων (relationship extraction). Κάθε οντότητα αναπαρίσταται ως κόμβος, ενώ οι συσχετίσεις μεταξύ οντοτήτων αποτυπώνονται ως ακμές με

συγκεκριμένο τύπο και ιδιότητες, ακολουθώντας την ιεραρχική αλυσίδα «υπόθεση → έγγραφο → οντότητα». Η γραφηματική αυτή προσέγγιση επιτρέπει τόσο την άμεση και ευέλικτη οπτικοποίηση της γνώσης όσο και την εφαρμογή προηγμένων αλγορίθμων ανάλυσης, όπως η ανίχνευση κοινοτήτων (community detection), ο υπολογισμός συντομότερων διαδρομών (shortest path algorithms) και η μέτρηση κεντρικότητας κόμβων (graph centrality metrics) [39]. Με αυτόν τον τρόπο, η Neo4j λειτουργεί ως κεντρικός κόμβος αναπαράστασης και αξιοποίησης της συσχετιζόμενης πληροφορίας, ενισχύοντας τη δυνατότητα ανακάλυψης κρυφών σχέσεων και προτύπων.

Η Neo4j συνεργάζεται συμπληρωματικά με το OpenSearch και την Qdrant: το OpenSearch εξυπηρετεί αναζητήσεις βάσει λέξεων-κλειδιών, η Qdrant καλύπτει σημασιολογική αναζήτηση, ενώ η Neo4j παρέχει τη δυνατότητα εξερεύνησης και ανάλυσης των διασυνδέσεων μεταξύ των δεδομένων. Έτσι, ο χρήστης μπορεί όχι μόνο να ανακτήσει πληροφορίες αλλά και να κατανοήσει τη δομή και τα μοτίβα που συνδέουν τα δεδομένα μεταξύ τους.

Η ενσωμάτωση της Neo4j στην εφαρμογή ενισχύει σημαντικά τη δυνατότητα αναπαράστασης και αξιοποίησης της παραγόμενης γνώσης, προσφέροντας ένα ισχυρό πλαίσιο για την ανάλυση περίπλοκων συσχετίσεων και καθιστώντας την αναπόσπαστο τμήμα της αρχιτεκτονικής του συστήματος [54], [40].

4.3 Ροή Επεξεργασίας μετά το Document Parsing – Εμπλουτισμός Δεδομένων

Μετά την αρχική εξαγωγή κειμένου και μεταδεδομένων από το **Apache Tika**, η ροή της εφαρμογής περνά στο στάδιο του εμπλουτισμού των δεδομένων μέσω προηγμένων τεχνικών επεξεργασίας φυσικής γλώσσας. Σε αυτό το στάδιο ενσωματώνονται δύο κρίσιμες τεχνολογίες:

- **spaCy**: Χρησιμοποιείται για την Αναγνώριση Οντοτήτων (*Named Entity Recognition – NER*) με πολυγλωσσικά μοντέλα. Η διαδικασία αυτή επιτρέπει την αυτόματη ανίχνευση και κατηγοριοποίηση οντοτήτων, όπως ονόματα, τοποθεσίες, ημερομηνίες και οργανισμοί, εμπλουτίζοντας το περιεχόμενο με σημασιολογικές πληροφορίες.
- **Ollama**: Χρησιμοποιείται για την παραγωγή διανυσματικών αναπαραστάσεων (*embeddings*) των εγγράφων και των οντοτήτων, με στόχο την υποστήριξη σημασιολογικής αναζήτησης μέσω της Qdrant. Η δυνατότητα εκτέλεσης LLM μοντέλων τοπικά ενισχύει την ιδιωτικότητα και επιτρέπει πλήρη έλεγχο της διαδικασίας.

Ο συνδυασμός των δύο τεχνολογιών επιτρέπει στο σύστημα να μετατρέπει απλό κείμενο σε πλούσια, δομημένη πληροφορία, έτοιμη να αξιοποιηθεί από τα υποσυστήματα αποθήκευσης και ανάκτησης δεδομένων που παρουσιάστηκαν στην προηγούμενη ενότητα.

4.3.1 Ενσωμάτωση του spaCy για Αναγνώριση Οντοτήτων (NER)

Η Αναγνώριση Οντοτήτων (*Named Entity Recognition – NER*) αποτελεί κρίσιμο υποσύστημα της εφαρμογής, καθώς επιτρέπει την αυτόματη ανάλυση του κειμένου που εξάγεται από τα μεταφορτωμένα έγγραφα και τον εντοπισμό πληροφοριών όπως ονόματα προσώπων, οργανισμοί, τοποθεσίες, ημερομηνίες, IP διευθύνσεις, domains και άλλα ειδικά αναγνωριστικά.

Σχήμα 4.5: Λογότυπο της spaCy

Πηγή: Official spaCy Website (https://spacy.io/styleguide)

Η βιβλιοθήκη **spaCy** είναι ένα ανοικτού κώδικα λογισμικό επεξεργασίας φυσικής γλώσσας (NLP) γραμμένο σε Python και Cython, που δημιουργήθηκε το 2015 από την εταιρεία Explosion AI με έδρα στο Βερολίνο. Έχει σχεδιαστεί ειδικά για εφαρμογές παραγωγικού επιπέδου, παρέχοντας υψηλή απόδοση και ακρίβεια, σε αντίθεση με πολλές άλλες NLP βιβλιοθήκες που επικεντρώνονται κυρίως στην ερευνητική χρήση. Το spaCy περιλαμβάνει προεκπαιδευμένα μοντέλα για πολλές γλώσσες και προσφέρει λειτουργίες όπως:

- Αναγνώριση οντοτήτων (NER)
- Ανάλυση γραμματικής δομής και εξαρτήσεων (*dependency parsing*)
- Μορφολογική ανάλυση και *lemmatization*
- Σημασιολογική αναπαράσταση κειμένου με διανυσματικούς χώρους

Από την πρώτη του έκδοση, το spaCy έχει καθιερωθεί ως ένα από τα πιο δημοφιλή και αξιόπιστα εργαλεία για εμπορικές εφαρμογές NLP, χάρη στην ταχύτητα εκτέλεσης, το καλά σχεδιασμένο API και την πλήρη τεκμηρίωση [55].

4.3.1.1 Εκπαιδευμένα μοντέλα spaCy για NER σε ελληνικά και αγγλικά κείμενα

Για την αναγνώριση οντοτήτων (Named Entity Recognition – NER) στην παρούσα εφαρμογή χρησιμοποιούνται τα επίσημα προκατασκευασμένα pipelines της βιβλιοθήκης **spaCy** (έκδοση 3.8.0). Τα μοντέλα αυτά, διανεμόμενα από την εταιρεία *Explosion*, διαθέτουν πλήρη τεκμηρίωση, ενεργή υποστήριξη και έχουν εκπαιδευτεί σε μεγάλα, καθαρά και ελεγμένα σύνολα δεδομένων, εξασφαλίζοντας σταθερή και προβλέψιμη απόδοση.

Τα ελληνικά pipelines (**el_core_news_sm**, **el_core_news_md**, **el_core_news_lg**) εκπαιδεύτηκαν κυρίως στο *UD Greek GDT v2.8* για συντακτική και μορφολογική ανάλυση, και στο *Greek NER Corpus* (Google Summer of Code 2018) για την εκπαίδευση του συστήματος αναγνώρισης οντοτήτων. Η θεματολογία τους προέρχεται κυρίως από ειδησεογραφικό λόγο, με αποτέλεσμα τα δεδομένα να είναι γραμματικά ορθά και θεματικά συνεκτικά, γεγονός που ευνοεί την απόδοση σε τυποποιημένα κείμενα αλλά περιορίζει τη γενίκευση σε άτυπα ή θορυβώδη δεδομένα, όπως αυτά που ενδέχεται να επεξεργάζεται η εφαρμογή [57].

Αντίστοιχα, τα αγγλικά pipelines (**en_core_web_sm**, **en_core_web_md**, **en_core_web_lg**, **en_core_web_trf**) έχουν εκπαιδευτεί κυρίως στο *OntoNotes 5*, ένα από τα πιο αναγνωρισμένα σύνολα δεδομένων για NLP, σε συνδυασμό με το *WordNet 3.0* και πρόσθετες πηγές (OSCAR, Wikipedia, OpenSubtitles, WMT News Crawl για τα μοντέλα με ενσωματωμένα διανύσματα).

Η χρήση μετασχηματιστικών μοντέλων (transformers) ή μεγάλων γλωσσικών μοντέλων (LLMs) απορρίφθηκε στην παρούσα φάση λόγω του υψηλού υπολογιστικού κόστους, της αυξημένης καθυστέρησης απόκρισης και της πολυπλοκότητας παραγωγικής ενσωμάτωσης, παρά την υψηλότερη ακρίβεια που μπορούν να προσφέρουν σε ελεγχόμενα περιβάλλοντα. Παράλληλα, οι καθαρά λεξικοκεντρικές ή βασισμένες σε κανόνες (regex-only) προσεγγίσεις κρίθηκαν ανεπαρκείς για την ποικιλομορφία και την ασάφεια των πραγματικών δεδομένων. Επιλέχθηκε, συνεπώς, μια υβριδική λύση βασισμένη στο spaCy, η οποία συνδυάζει αξιόπιστα και τεκμηριωμένα pipelines με στοχευμένες regex για ειδικές κατηγορίες οντοτήτων, προσφέροντας ισορροπία ανάμεσα σε ακρίβεια, ταχύτητα και παραγωγική σταθερότητα. Σε αντίθεση με τα LLMs, η προσέγγιση αυτή έχει ντετερμινιστικό χαρακτήρα – η ίδια είσοδος παράγει πάντοτε το ίδιο αποτέλεσμα – διευκολύνοντας έτσι τον έλεγχο, την αναπαραγωγιμότητα και την τεκμηρίωση των διαδικασιών[58].
Ο πίνακας που ακολουθεί συνοψίζει τα διαθέσιμα μοντέλα και τις μετρικές NER:

Πίνακας 4.1: Μετρικές ακρίβειας NER για τα προκατασκευασμένα μοντέλα spaCy (έκδοση 3.8.0)

Α/Α	**Γλώσσα**	**Μοντέλο**	**ENT Precision**	**ENT Recall**	**ENT F1**
1	Ελληνικά	el_core_news_sm	0.71	0.68	0.70
2	Ελληνικά	el_core_news_md	0.75	0.76	0.76
3	Ελληνικά	el_core_news_lg	0.76	0.78	0.77
4	Αγγλικά	en_core_web_sm	0.84	0.84	0.84
5	Αγγλικά	en_core_web_md	0.84	0.85	0.85
6	Αγγλικά	en_core_web_lg	0.85	0.86	0.86
7	Αγγλικά	en_core_web_trf	0.90	0.90	0.90

4.3.1.2 Υλοποίηση στην εφαρμογή

Στην παρούσα εφαρμογή, το spaCy αξιοποιείται για την αναγνώριση οντοτήτων σε συνδυασμό με τεχνικές Regular Expressions (Regex), οι οποίες συμπληρώνουν την αναγνώριση εξειδικευμένων τύπων πληροφορίας. Χρησιμοποιούνται τα προεκπαιδευμένα μοντέλα el_core_news_sm για τα ελληνικά και en_core_web_sm για τα αγγλικά, επιτυγχάνοντας ισορροπία μεταξύ ακρίβειας και ταχύτητας.
Η υλοποίηση ακολουθεί την αρχιτεκτονική **microservice**, με το NER να ενσωματώνεται ως αυτόνομη υπηρεσία βασισμένη στο **FastAPI**. Το FastAPI, ένα σύγχρονο Python framework για ανάπτυξη RESTful APIs, παρέχει υψηλή απόδοση, αυτόματη τεκμηρίωση και εύκολη ενσωμάτωση με άλλα υποσυστήματα. Η υπηρεσία nlp_ner (docker service) δέχεται κείμενο, εκτελεί την ανάλυση με spaCy και επιστρέφει τις αναγνωρισμένες οντότητες σε μορφή JSON [56].
Οι προκαθορισμένες δυνατότητες του spaCy εμπλουτίζονται με **custom Regex κανόνες**, οι οποίοι επιτρέπουν την αναγνώριση IP διευθύνσεων, email, domains, IBANs και διευθύνσεων κρυπτονομισμάτων. Οι κανόνες αυτοί είναι παραμετροποιήσιμοι, ώστε να καλύπτουν ανάγκες ανά τύπο εγγράφου ή σενάριο χρήσης.
Το σύστημα διατηρεί τόσο τις οντότητες που παράγονται από τα μοντέλα NER όσο και τα δεδομένα που επιβεβαιώθηκαν ή απορρίφθηκαν από τον χρήστη. Η πληροφορία αυτή, σε συνδυασμό με το ιστορικό αλλαγών, δημιουργεί ένα πολύτιμο σύνολο δεδομένων που μπορεί να αξιοποιηθεί για μελλοντικό *fine-tuning* των μοντέλων, προσαρμόζοντάς τα στις ειδικές ανάγκες και το λεξιλόγιο του

εκάστοτε οργανισμού. Με αυτόν τον τρόπο, το σύστημα αποκτά τη δυνατότητα συνεχούς βελτίωσης της ακρίβειας και της συνάφειας των αναγνωρίσεων.
Η συνδυαστική χρήση spaCy, Regex και FastAPI παρέχει ένα ευέλικτο και αποδοτικό πλαίσιο αναγνώρισης οντοτήτων, κατάλληλο για απαιτητικά περιβάλλοντα παραγωγής, με δυνατότητα εύκολης επέκτασης και προσαρμογής σε νέους τύπους δεδομένων [56].

4.3.2 Ενσωμάτωση του Ollama για Τοπική Εκτέλεση Μεγάλων Γλωσσικών Μοντέλων (LLMs)

Η πλατφόρμα **Ollama** ενσωματώθηκε στην εφαρμογή ως μηχανισμός τοπικής εκτέλεσης μεγάλων γλωσσικών μοντέλων (*Large Language Models – LLMs*), προσφέροντας δυνατότητες που υπερβαίνουν την παραγωγή διανυσματικών αναπαραστάσεων (*embeddings*). Μέσω του Ollama, το σύστημα μπορεί να φορτώνει, να διαχειρίζεται και να εκτελεί μοντέλα τελευταίας γενιάς, επιτρέποντας την ανάπτυξη λειτουργιών όπως αυτόματη παραγωγή κειμένου, περίληψη εγγράφων, απάντηση σε ερωτήματα, καθώς και δημιουργία embeddings για σημασιολογική αναζήτηση [59].

Σχήμα 4.6: Λογότυπο του Ollama

Πηγή: Official Ollama Website (https://ollama.com)

Το Ollama είναι μια πλατφόρμα ανοικτού κώδικα που επιτρέπει την εγκατάσταση και εκτέλεση LLMs τοπικά, σε περιβάλλοντα με CPU ή GPU, χωρίς εξάρτηση από εξωτερικές cloud υπηρεσίες. Υποστηρίζει δημοφιλή μοντέλα όπως LLaMA, Mistral, Vicuna, Gemma και άλλα, παρέχοντας ένα ενοποιημένο API για αλληλεπίδραση είτε μέσω REST κλήσεων είτε μέσω CLI. Η τοπική εκτέλεση διασφαλίζει:

- **Απόλυτη ιδιωτικότητα** των δεδομένων, καθώς δεν μεταφέρονται σε τρίτους.
- **Πλήρη έλεγχο** στην επιλογή και παραμετροποίηση μοντέλων.
- **Δυνατότητα fine-tuning** για εξειδικευμένες ανάγκες ενός οργανισμού.
- **Συμμόρφωση με νομικές και κανονιστικές απαιτήσεις** για την προστασία ευαίσθητων πληροφοριών [60].

4.3.2.1 Υλοποίηση στην εφαρμογή

Στο πλαίσιο της εφαρμογής, το Ollama αξιοποιείται για δύο κύριες λειτουργίες:

1. Δημιουργία διανυσματικών αναπαραστάσεων (embeddings) από έγγραφα, οι οποίες αποθηκεύονται στη Qdrant και χρησιμοποιούνται για σημασιολογική αναζήτηση.

2. Εκτέλεση LLM ερωτήσεων και παραγωγής απαντήσεων, υποστηρίζοντας λειτουργίες όπως δημιουργία συνόψεων και αυτόματη ανάλυση περιεχομένου.

Για την πρώτη λειτουργία επιλέχθηκε το μοντέλο **nomic-embed-text**, το οποίο έχει βελτιστοποιηθεί για τη δημιουργία υψηλής ποιότητας κειμενικών embeddings. Το μοντέλο προσφέρει ισορροπία μεταξύ ακρίβειας στις σημασιολογικές συσχετίσεις και ταχύτητας εκτέλεσης, ενώ μπορεί να λειτουργήσει αποδοτικά σε τοπικό περιβάλλον χωρίς σημαντικές απαιτήσεις σε υπολογιστικούς πόρους.

Για τη δεύτερη λειτουργία χρησιμοποιείται το μοντέλο **gemma3:1b**, ένα ελαφρύ αλλά ικανό LLM που υποστηρίζει βασικές διεργασίες κατανόησης και παραγωγής κειμένου με μικρό αποτύπωμα μνήμης και γρήγορη απόκριση. Η επιλογή του βασίστηκε στους περιορισμούς των διαθέσιμων πόρων και στην ανάγκη για άμεση εκτέλεση ερωτημάτων σε πραγματικό χρόνο. Ωστόσο, η αρχιτεκτονική του συστήματος δεν περιορίζει τον χρήστη — μπορεί να αξιοποιήσει οποιοδήποτε άλλο εγκατεστημένο μοντέλο του Ollama, ανάλογα με τις ανάγκες και τις δυνατότητες του περιβάλλοντος. Η επιλογή και η εναλλαγή μοντέλων γίνεται απευθείας από τη διεπαφή χρήστη (UI), προσφέροντας ευελιξία και δυνατότητα πειραματισμού χωρίς τεχνικές παρεμβάσεις στον πυρήνα της εφαρμογής.

Η αρχιτεκτονική που ακολουθείται επιτρέπει την εναλλαγή του Ollama με εμπορικές πλατφόρμες LLM (π.χ. OpenAI, Anthropic), προσφέροντας ευελιξία στην προσαρμογή σε διαφορετικές επιχειρησιακές και τεχνολογικές απαιτήσεις, χωρίς εξάρτηση από έναν μόνο πάροχο.

4.3.3 Επεξεργασία Εγγράφων και Pipeline Orchestration

Το υποσύστημα επεξεργασίας εγγράφων αποτελεί το κεντρικό σημείο σύνδεσης ανάμεσα στο στάδιο της αρχικής εξαγωγής περιεχομένου (Apache Tika) και στις υπηρεσίες σημασιολογικής ανάλυσης, αναγνώρισης οντοτήτων και εξαγωγής γνώσης. Η κλάση ParsedEventHandler και η συνοδευτική ακολουθία Celery tasks λειτουργούν ως ο ενορχηστρωτής της ροής, εξασφαλίζοντας ότι κάθε έγγραφο περνά από τα απαραίτητα στάδια εμπλουτισμού, αποθήκευσης και αναζήτησης, με πλήρη αξιοποίηση των διαθέσιμων υποσυστημάτων της εφαρμογής.

Η διαδικασία ξεκινά με την ανίχνευση γεγονότων στο Redis, τα οποία αφορούν την ολοκλήρωση του parsing εγγράφων. Στη συνέχεια, το αντίστοιχο περιεχόμενο ανακτάται από το MinIO, επικυρώνεται και προετοιμάζεται για τις επόμενες φάσεις. Κατά την προετοιμασία γίνεται εξαγωγή και κανονικοποίηση μεταδεδομένων όπως ο τύπος αρχείου, η κωδικοποίηση και η γλώσσα, καθώς και καθαρισμός του κειμένου από περιττά κενά και ειδικούς χαρακτήρες. Το περιεχόμενο στη συνέχεια τεμαχίζεται σε επιμέρους τμήματα (chunks) με βάση το πλήθος των tokens, ώστε να μπορεί να επεξεργαστεί αποδοτικά στα επόμενα βήματα.

Ακολουθεί η εκτέλεση του Named Entity Recognition (NER) μέσω της υπηρεσίας NERService, η οποία επικοινωνεί με μικροϋπηρεσία FastAPI που ενσωματώνει τη βιβλιοθήκη spaCy με τα προεκπαιδευμένα μοντέλα el_core_news_sm για τα ελληνικά και en_core_web_sm για τα αγγλικά. Παράλληλα, χρησιμοποιούνται κανονικές εκφράσεις (Regex) για την αναγνώριση εξειδικευμένων τύπων οντοτήτων. Τα αποτελέσματα αποθηκεύονται στη σχεσιακή βάση PostgreSQL, ενώ παρέχεται στον χρήστη η δυνατότητα επιβεβαίωσης ή διόρθωσης μέσω του γραφικού περιβάλλοντος, αυξάνοντας την ακρίβεια και παρέχοντας δεδομένα για πιθανό μελλοντικό fine-tuning των μοντέλων.

Η επόμενη φάση αφορά τη δημιουργία διανυσματικών αναπαραστάσεων του περιεχομένου μέσω της υπηρεσίας EmbeddingService. Το κείμενο που έχει τεμαχιστεί σε chunks μετατρέπεται σε διανύσματα (embeddings) τα οποία αποθηκεύονται στη βάση Qdrant, καθιστώντας δυνατή τη σημασιολογική αναζήτηση και την ευφυή ανάκτηση πληροφορίας.

Στη φάση της εξαγωγής γνώσης, το περιεχόμενο υποβάλλεται σε επεξεργασία από την υπηρεσία KnowledgeGraphService για την παραγωγή σχέσεων τύπου subject–predicate–object. Εφαρμόζεται η

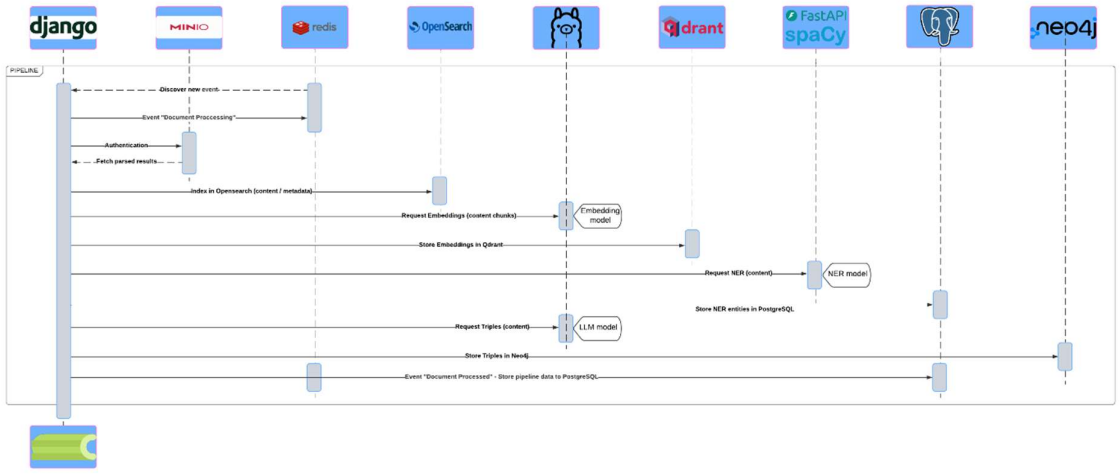
μέθοδος heuristic_extract_triples, η οποία βασίζεται σε λεκτικά μοτίβα, συντακτική ανάλυση και κανόνες, και οι προκύπτουσες τριάδες αποθηκεύονται στη βάση Neo4j. Η επιλογή αυτής της μεθόδου έναντι της χρήσης LLMs έγινε λόγω του χαμηλότερου υπολογιστικού κόστους, χωρίς να αποκλείεται η μελλοντική αξιοποίηση μεγάλων γλωσσικών μοντέλων.

Μετά την ολοκλήρωση όλων των βημάτων, η κατάσταση του εγγράφου ενημερώνεται μέσω της DocumentStatusService και τα σχετικά γεγονότα μπορούν να αποσταλούν μέσω WebSocket καναλιών στους χρήστες. Το υποσύστημα ParsedEventHandler λειτουργεί ως κεντρικός μηχανισμός που μετατρέπει αδόμητο περιεχόμενο σε δομημένη, αναζητήσιμη και σημασιολογικά εμπλουτισμένη γνώση, ενοποιώντας πλήθος υπηρεσιών της αρχιτεκτονικής και επιτρέποντας ένα υψηλό επίπεδο ευφυούς ανάλυσης.

Η συνολική ροή του υποσυστήματος δεν είναι αυστηρά σειριακή· αντίθετα, βασίζεται σε ασύγχρονη ενορχήστρωση διεργασιών, όπου πολλά στάδια μπορούν να εκτελούνται παράλληλα και να ολοκληρώνονται ανεξάρτητα. Ο χρήστης έχει συνεχή εικόνα της προόδου μέσω του γραφικού περιβάλλοντος, χωρίς να απαιτείται η ολοκλήρωση όλων των εργασιών. Η ροή αυτή αποτυπώνεται στο παρακάτω διάγραμμα ακολουθιών.

4.3.4 Διάγραμμα Ακολουθιών Επεξεργασίας Εγγράφων

Το διάγραμμα ακολουθιών (Σχήμα 4.7) παρουσιάζει τα βήματα επεξεργασίας και σημασιολογικής ανάκτησης εγγράφων μετά την ολοκλήρωση της ανάλυσης από το Apache Tika. Στο διάγραμμα φαίνονται οι κύριες αλληλεπιδράσεις μεταξύ των υποσυστημάτων (Django, Redis, MinIO, OpenSearch, Qdrant, spaCy, Neo4j κ.λπ.) καθώς και τα στάδια εμπλουτισμού, αποθήκευσης και ενημέρωσης του χρήστη.



Σχήμα 4.7: Ροή Επεξεργασίας Εγγράφων

Django → Redis

Μετά την ολοκλήρωση της ανάλυσης του εγγράφου από το Apache Tika (Parsing και OCR), η υπηρεσία ParsedEventHandler στο backend Django ανιχνεύει το νέο parsed αρχείο και δημιουργεί event τύπου "Document Parsed", το οποίο τοποθετείται στο Redis μέσω Celery, ενεργοποιώντας τις επόμενες εργασίες.

MinIO → Django

Το επεξεργασμένο αρχείο .json ανακτάται από τον MinIO (object storage) μέσω της υπηρεσίας MinIOService, χρησιμοποιώντας το κατάλληλο bucket (parsed-documents), presigned URL και credentials. Στη συνέχεια γίνεται normalization του κειμένου και προετοιμασία για εμπλουτισμό και indexing.

Indexing → OpenSearch

Το κανονικοποιημένο περιεχόμενο, μαζί με τα μεταδεδομένα (π.χ. όνομα αρχείου, ημερομηνία, MIME type), αποστέλλεται στην υπηρεσία OpenSearchService και αποθηκεύεται σε index. Έτσι υποστηρίζεται ταχεία και ευέλικτη αναζήτηση με keywords, filters και χρονικά κριτήρια.

Request Embeddings → Qdrant

Το κείμενο τεμαχίζεται σε chunks με χρήση TokenTextSplitter και αποστέλλεται στην υπηρεσία EmbeddingService για παραγωγή embeddings. Τα αποτελέσματα αποθηκεύονται στο Qdrant μέσω QdrantVectorStorageService, επιτρέποντας σημασιολογική αναζήτηση.

NER → spaCy (FastAPI)

Το ίδιο κείμενο αποστέλλεται στην υπηρεσία NERService (FastAPI), η οποία βασίζεται στη βιβλιοθήκη spaCy με τα προεκπαιδευμένα μοντέλα el_core_news_sm και en_core_web_sm. Η έξοδος περιλαμβάνει αναγνωρισμένες οντότητες (PERSON, ORG, LOC κ.λπ.).

Αίτηση για Εξαγωγή Σχέσεων (Triples)

Το περιεχόμενο υποβάλλεται στην υπηρεσία KnowledgeGraphService για εξαγωγή σχέσεων τύπου (οντότητα1 – σχέση – οντότητα2). Όταν δεν χρησιμοποιείται LLM λόγω κόστους, εφαρμόζεται η μέθοδος heuristic_extract_triples με λεκτικά μοτίβα, dependency parsing και κανόνες.

Αποθήκευση → Neo4j

Οι παραγόμενες τριπλέτες καταχωρούνται στο Neo4j ως γράφος γνώσης, υποστηρίζοντας ανάλυση, συσχετίσεις και εξαγωγή συμπερασμάτων με βάση τις σχέσεις των οντοτήτων.

Επαλήθευση Χρήστη

Οι οντότητες που προκύπτουν από το NER αποθηκεύονται προσωρινά και είναι διαθέσιμες στο UI, όπου ο χρήστης μπορεί να τις επιβεβαιώσει ή να τις διορθώσει. Αυτή η παρέμβαση βελτιώνει την ακρίβεια και δημιουργεί δεδομένα για πιθανό fine-tuning μοντέλων στο μέλλον.

Αποθήκευση → PostgreSQL / Neo4j

Οι επιβεβαιωμένες οντότητες αποθηκεύονται μόνιμα στην PostgreSQL (για διαχειριστική και αναλυτική χρήση) και στο Neo4j (για γραφοκεντρική ανάλυση).

Συνεχής Ενημέρωση Προόδου

Η εφαρμογή ενημερώνει σε πραγματικό χρόνο την κατάσταση κάθε σταδίου επεξεργασίας (π.χ. indexed, embedded, ner_completed, tripled) και καταγράφει όλα τα ενδιάμεσα και τελικά αποτελέσματα στη PostgreSQL. Ο χρήστης μπορεί ανά πάσα στιγμή να δει την πρόοδο μέσω του UI, χωρίς να περιμένει την ολοκλήρωση όλων των εργασιών.

4.4 Επίλογος

Το δεύτερο στάδιο ανάπτυξης επικεντρώθηκε στην υλοποίηση μηχανισμών αναζήτησης και ανάκτησης πληροφορίας, αξιοποιώντας τα δομημένα δεδομένα που προέκυψαν από το προηγούμενο στάδιο επεξεργασίας εγγράφων. Μέσα από την ενσωμάτωση εξειδικευμένων εργαλείων επεξεργασίας φυσικής γλώσσας και την εφαρμογή τεχνικών ευρετηρίασης, η εφαρμογή απέκτησε τη δυνατότητα να παρέχει γρήγορα και ακριβή αποτελέσματα, ακόμη και σε περιβάλλοντα με μεγάλο όγκο δεδομένων.

Η έμφαση δόθηκε στη βελτιστοποίηση της εμπειρίας χρήστη, εξασφαλίζοντας αφενός την ευελιξία των ερωτημάτων και αφετέρου την ακρίβεια και συνάφεια των αποτελεσμάτων. Παράλληλα, υιοθετήθηκαν

πρακτικές που επιτρέπουν την κλιμάκωση της λειτουργικότητας, ώστε το σύστημα να παραμένει αποδοτικό σε σενάρια αυξημένου φόρτου.

Με την ολοκλήρωση του σταδίου αυτού, η εφαρμογή διαθέτει πλέον μία σταθερή και αποτελεσματική υποδομή αναζήτησης, ικανή να αξιοποιεί πλήρως το περιεχόμενο που έχει εξαχθεί. Στο επόμενο κεφάλαιο, η εστίαση μετατοπίζεται στην παρουσίαση, οπτικοποίηση και ανάλυση της πληροφορίας, με στόχο τη μετατροπή των δεδομένων σε χρήσιμη γνώση για τον τελικό χρήστη.

Κεφάλαιο 5ο: Από την Εισαγωγή Δεδομένων έως την Υβριδική Αναζήτηση

5.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται οι τελικές λειτουργίες της εφαρμογής, όπως αυτές γίνονται αντιληπτές από την πλευρά του χρήστη. Η έμφαση δίνεται στη ροή από την αρχική είσοδο στο σύστημα έως την αναζήτηση και αξιοποίηση της πληροφορίας, αναδεικνύοντας την αλληλεπίδραση μεταξύ των διαφόρων υποσυστημάτων που έχουν αναπτυχθεί. Ειδικότερα, περιγράφονται η διαδικασία σύνδεσης και ταυτοποίησης χρηστών, η μεταφόρτωση και επεξεργασία εγγράφων, η αναγνώριση οντοτήτων, η απεικόνιση μέσω γράφων, καθώς και οι μηχανισμοί υβριδικής αναζήτησης και επεξεργασίας ερωτημάτων με LLM. Μέσα από στιγμιότυπα οθόνης και επεξηγηματικά παραδείγματα αναδεικνύεται πώς το σύστημα μετατρέπει τα αδόμητα δεδομένα σε αξιοποιήσιμη γνώση, προσφέροντας στον χρήστη δυνατότητες που συνδυάζουν την παραδοσιακή αναζήτηση με τη δύναμη της σημασιολογικής επεξεργασίας.

5.2 Πρόσβαση και Ασφάλεια

5.2.1 Σύνδεση Χρηστών και Ρόλοι

Η πρόσβαση στην εφαρμογή γίνεται μέσω ενός ασφαλούς μηχανισμού σύνδεσης, ο οποίος εξασφαλίζει ότι κάθε χρήστης ταυτοποιείται με μοναδικά διαπιστευτήρια και αποκτά πρόσβαση αποκλειστικά στις λειτουργίες που αντιστοιχούν στον ρόλο του. Η διαδικασία αυτή στηρίζεται σε σύγχρονες πρακτικές ασφάλειας, με κρυπτογραφημένη επικοινωνία, αποθήκευση κωδικών σε κατακερματισμένη μορφή και δυνατότητα χρήσης δεύτερου παράγοντα ταυτοποίησης (Two-Factor Authentication – 2FA).
Οι χρήστες οργανώνονται σε τέσσερις βασικούς ρόλους, οι οποίοι αντικατοπτρίζουν την υφιστάμενη ιεραρχία ενός οργανισμού και καθορίζουν το εύρος πρόσβασης στην πληροφορία:

- **Basic User**: Έχει περιορισμένη πρόσβαση και μπορεί να δει μόνο τις υποθέσεις που του έχουν ανατεθεί ή τα έγγραφα που έχει ανεβάσει (upload).
- **Supervisor (Επόπτης)**: Έχει πρόσβαση σε όλες τις υποθέσεις και τα έγγραφα του τμήματός του, επιβλέποντας το έργο των βασικών χρηστών (Basic Users).
- **Head (Διευθυντής)**: Βλέπει τις υποθέσεις όλων των τμημάτων που υπάγονται στη Διεύθυνσή του, έχοντας εποπτεία σε ανώτερο επίπεδο.
- **Administrator (Διαχειριστής)**: Διαθέτει πλήρη πρόσβαση στο σύστημα, με δυνατότητα διαχείρισης υποθέσεων, χρηστών και οργανωτικών μονάδων.

Η ύπαρξη αυτής της διαβάθμισης διασφαλίζει ότι κάθε χρήστης βλέπει μόνο τις πληροφορίες που είναι απαραίτητες για τον ρόλο του, μειώνοντας τον κίνδυνο τυχαίας ή μη εξουσιοδοτημένης πρόσβασης. Παράλληλα, ο μηχανισμός αυτός ενισχύει τη διαφάνεια και τη λογοδοσία, καθώς οι ευθύνες και τα δικαιώματα είναι σαφώς καθορισμένα.
Η διαχείριση των χρηστών και των ρόλων πραγματοποιείται μέσα από το περιβάλλον διαχείρισης της εφαρμογής, όπου ο διαχειριστής μπορεί να παρακολουθεί εγγεγραμμένους χρήστες, να καθορίζει τα δικαιώματα κάθε ρόλου και να διασφαλίζει τη σωστή αντιστοίχιση με τις οργανωτικές μονάδες (Διεύθυνση, Τμήμα). Η δυνατότητα αυτή καθιστά το σύστημα ευέλικτο και προσαρμόσιμο σε πραγματικές οργανωτικές δομές, ενώ επιτρέπει την εφαρμογή πολιτικών ασφαλείας, όπως η υποχρεωτική αλλαγή κωδικού ή η ενεργοποίηση 2FA.

5.2.2 Ασφάλεια Λογαριασμών και Δεδομένων

Η ασφάλεια αποτελεί θεμελιώδη παράμετρο του σχεδιασμού της εφαρμογής, καθώς το σύστημα χειρίζεται δεδομένα που μπορεί να είναι ευαίσθητα ή ακόμη και απόρρητα. Για τον σκοπό αυτό έχουν υιοθετηθεί πολλαπλές δικλείδες προστασίας, τόσο σε επίπεδο αυθεντικοποίησης όσο και σε επίπεδο διαχείρισης δεδομένων.

Η διαδικασία πιστοποίησης των χρηστών βασίζεται σε ασφαλείς μηχανισμούς ταυτοποίησης (authentication) και εξουσιοδότησης (authorization). Η είσοδος στο σύστημα πραγματοποιείται μέσω μοναδικών διαπιστευτηρίων, ενώ όλα τα συνθηματικά αποθηκεύονται σε κατακερματισμένη μορφή, ώστε να αποτρέπεται η ανάκτησή τους ακόμη και σε περίπτωση παραβίασης της βάσης δεδομένων. Επιπλέον, η επικοινωνία μεταξύ χρήστη και συστήματος προστατεύεται με κρυπτογράφηση (HTTPS/SSL), εξασφαλίζοντας την ακεραιότητα των δεδομένων κατά τη μεταφορά.

Ιδιαίτερη έμφαση δίνεται στην ενίσχυση της ασφάλειας μέσω πρόσθετων μηχανισμών. Ο χρήστης μπορεί να ενεργοποιήσει δεύτερο παράγοντα ταυτοποίησης (Two-Factor Authentication – 2FA), ώστε η είσοδος να απαιτεί και έναν επιπλέον κωδικό επαλήθευσης, αυξάνοντας σημαντικά το επίπεδο προστασίας του λογαριασμού. Επίσης εφαρμόζονται πολιτικές ασφαλείας για τη δημιουργία κωδικών, οι οποίες αποτρέπουν τη χρήση απλών ή εύκολα προβλέψιμων συνδυασμών, απαιτούν ελάχιστο μήκος και λαμβάνουν υπόψη την αποφυγή ομοιότητας με προσωπικά δεδομένα του χρήστη.

Η πρόσβαση σε δεδομένα και λειτουργίες καθορίζεται αυστηρά από τον ρόλο κάθε χρήστη, ακολουθώντας το μοντέλο ελέγχου πρόσβασης βάσει ρόλων (RBAC). Με αυτόν τον τρόπο, οι βασικοί χρήστες περιορίζονται σε προσωπικά δεδομένα και υποθέσεις, οι επόπτες και οι διευθυντές έχουν πρόσβαση σε ευρύτερες οργανωτικές ενότητες, ενώ ο διαχειριστής διατηρεί πλήρη έλεγχο σε όλο το σύστημα. Η εξουσιοδότηση αυτή εφαρμόζεται σε κάθε κρίσιμη ενέργεια, διασφαλίζοντας ότι ο χρήστης δεν μπορεί να εκτελέσει πράξεις πέραν των αρμοδιοτήτων του.

Συνολικά, το υποσύστημα ασφάλειας λειτουργεί ως η «πύλη προστασίας» της εφαρμογής, εγγυώμενο ότι μόνο εξουσιοδοτημένοι χρήστες αποκτούν πρόσβαση και ότι η πληροφορία παραμένει ασφαλής σε όλα τα στάδια χρήσης. Με αυτόν τον τρόπο, ενισχύεται η εμπιστοσύνη των χρηστών και δημιουργείται ένα αξιόπιστο περιβάλλον διαχείρισης δεδομένων, συμβατό με τις καλές πρακτικές και τα διεθνή πρότυπα ασφάλειας.

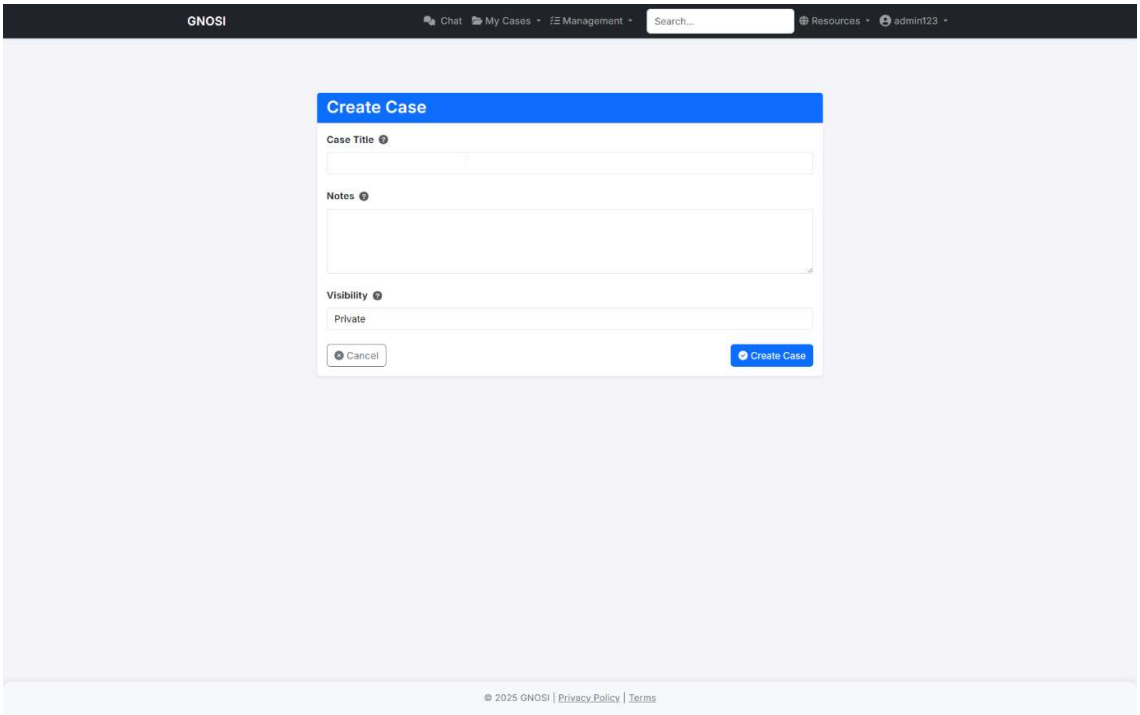
5.3 Διαχείριση Υποθέσεων και Εγγράφων

5.3.1 Δημιουργία Υπόθεσης και πλαισίωση της πληροφορίας

Πριν από κάθε ενέργεια μεταφόρτωσης, ο χρήστης δημιουργεί την υπόθεση (Case) που θα λειτουργήσει ως οργανωτικό και νοηματικό πλαίσιο για τα επόμενα τεκμήρια. Η δημιουργία της υπόθεσης (Case) πραγματοποιείται μέσω μιας καθοδηγούμενης φόρμας με σαφείς ετικέτες και επεξηγηματικά βοηθήματα, ώστε ο τίτλος, οι σημειώσεις και οι βασικές ιδιότητες να αποδοθούν με συνέπεια. Ο τίτλος της υπόθεσης ακολουθεί προκαθορισμένο πρότυπο καταγραφής (pattern), το οποίο λειτουργεί ως μηχανισμός ελέγχου συνέπειας και διευκολύνει τόσο την ταξινόμηση όσο και την αναζήτηση στο μέλλον. Με την οριστικοποίηση, η υπόθεση συνδέεται αυτόματα με την οργανωτική θέση του χρήστη (τμήμα και διεύθυνση), ενώ η ορατότητα ορίζεται με βάση τις ισχύουσες πολιτικές (ιδιωτική, τοπική, δημόσια), εξασφαλίζοντας ότι η πρόσβαση θα είναι ευθυγραμμισμένη με τον ρόλο και τις αρμοδιότητες κάθε χρήστη.

Η επιλογή να προηγείται η δημιουργία της υπόθεσης (Case) δεν είναι τυπική διαδικασία, αλλά ουσιώδης πράξη πλαισίωσης της πληροφορίας: τα έγγραφα που θα ακολουθήσουν εγγράφονται εξαρχής σε

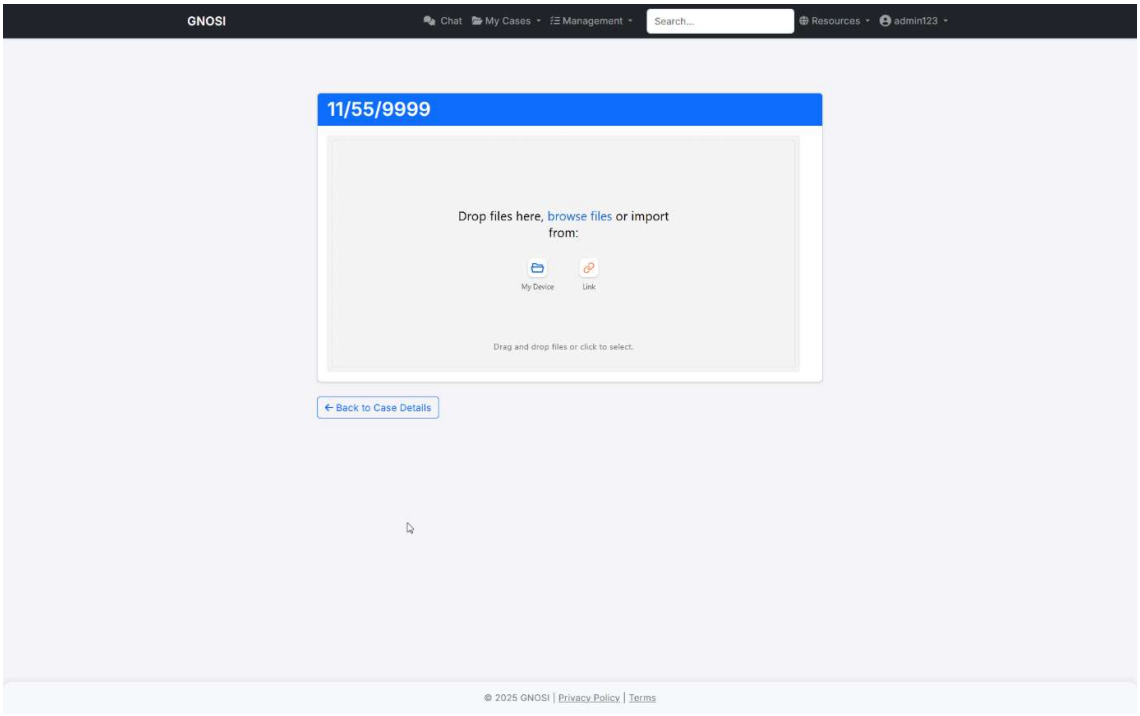
συγκεκριμένο διοικητικό και επιχειρησιακό πλαίσιο, αποφεύγονται ασάφειες προέλευσης και ενισχύεται η ιχνηλασιμότητα. Αμέσως μετά τη δημιουργία, ο χρήστης μεταβαίνει στην αναλυτική προβολή της υπόθεσης, όπου είναι διαθέσιμες οι βασικές λειτουργίες: επισκόπηση μεταδεδομένων, προσθήκη εγγράφων, παρακολούθηση κατάστασης επεξεργασίας και —σε μεταγενέστερα στάδια— αξιοποίηση γραφημάτων και μηχανισμών αναζήτησης. Με αυτόν τον τρόπο, το σύστημα επιβάλλει μια φυσική ροή εργασίας (Case → Document) που εναρμονίζεται με την οργανωτική πρακτική και αποτρέπει αποσπασματικές καταχωρίσεις εκτός πλαισίου.



Σχήμα 5.1: Φόρμα δημιουργίας υπόθεσης (Case). Ο χρήστης συμπληρώνει τίτλο με προκαθορισμένο πρότυπο, σημειώσεις και βασικές ιδιότητες ορατότητας.

5.4 Μεταφόρτωση και Αρχικοποίηση Επεξεργασίας Εγγράφων

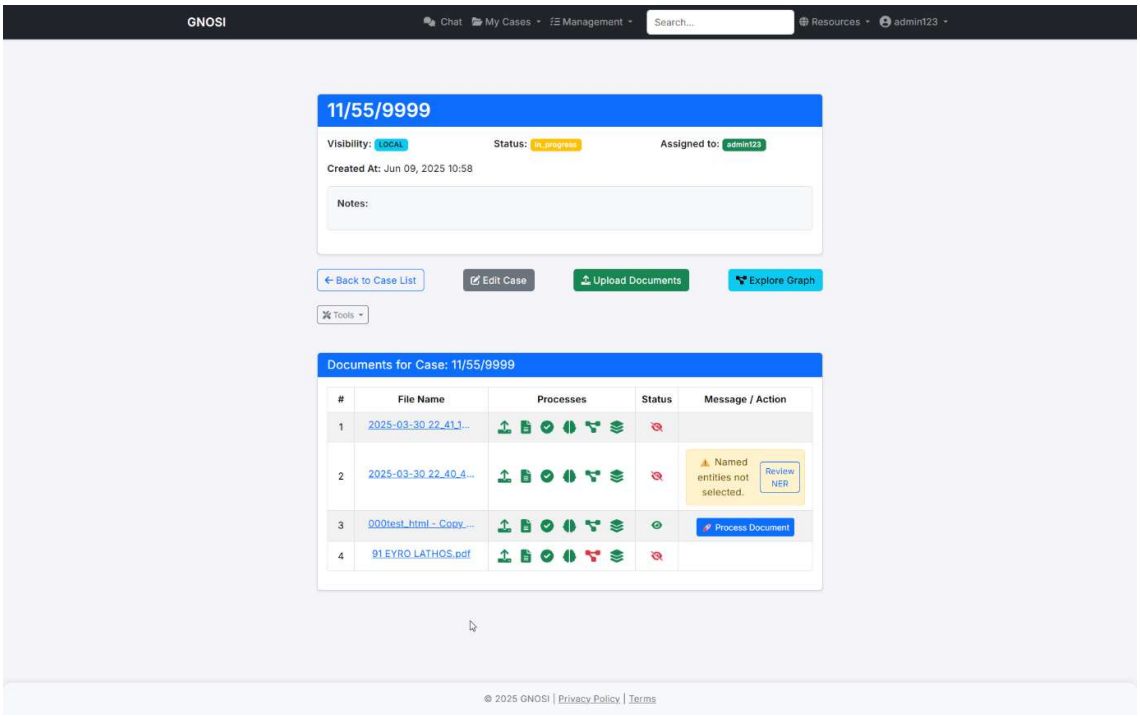
Η εισαγωγή εγγράφων αποτελεί το σημείο εκκίνησης του κύκλου ζωής της πληροφορίας: από την απόθεση αρχείων σε μια συγκεκριμένη υπόθεση (Case), έως την αυτοματοποιημένη επεξεργασία και την ένταξή τους στους μηχανισμούς αναζήτησης. Η εφαρμογή προσφέρει μια ενιαία εμπειρία μεταφόρτωσης που συνδυάζει ευχρηστία, ασφάλεια και άμεση ιχνηλασιμότητα, είτε τα αρχεία προέρχονται από τον τοπικό σταθμό εργασίας είτε από απομακρυσμένες πηγές (URL import).
Η οθόνη μεταφόρτωσης είναι ρητά δεσμευμένη σε μία υπόθεση, ώστε ο χρήστης να βλέπει πάντοτε τον τίτλο της υπόθεσης (Case) και να γνωρίζει το πλαίσιο στο οποίο εργάζεται. Η διαδικασία γίνεται με «σύρσιμο και απόθεση» ή με επιλογή αρχείων από τον δίσκο, ενώ η διεπαφή παρουσιάζει ξεκάθαρα την πρόοδο ανά αρχείο και υποστηρίζει πολλαπλές ταυτόχρονες μεταφορτώσεις. Μετά το πέρας, εμφανίζεται άμεση ειδοποίηση επιτυχίας ή αποτυχίας και ο χρήστης ανακατευθύνεται αυτόματα στην προβολή της υπόθεσης, όπου τα νέα Document καταγράφονται με την τρέχουσα κατάστασή τους (π.χ. uploaded, processing, parsed).



Σχήμα 5.2: Οθόνη μεταφόρτωσης εγγράφων δεσμευμένη σε συγκεκριμένη υπόθεση (Case). Υποστήριξη πολλαπλών αρχείων, ένδειξη προόδου και άμεση ανατροφοδότηση.

Η μεταφορά αρχείων υλοποιείται με προ-εξουσιοδοτημένες διευθύνσεις (presigned URLs) προς τον αποθηκευτικό χώρο αντικειμένων, ώστε να αποφεύγεται η έκθεση ευαίσθητων διαπιστευτηρίων και να μειώνεται το φορτίο στο backend. Παράλληλα, για κάθε αρχείο καταχωρίζονται μεταδεδομένα που το συνδέουν μονοσήμαντα με την υπόθεση: ονομασία, εσωτερική διαδρομή αποθήκευσης, χρόνος και χρήστης μεταφόρτωσης. Η ονοματοδοσία ακολουθεί συνεπή λογική (CaseTitle/relativePath/filename), επιτρέποντας διαρκή έλεγχο προέλευσης και εύκολη αναζήτηση στο μέλλον.
Πέρα από το κλασικό τοπικό ανέβασμα, η εφαρμογή υποστηρίζει και εισαγωγή αρχείων απευθείας από απομακρυσμένες πηγές μέσω ειδικής συνοδευτικής υπηρεσίας. Η διαδικασία περιλαμβάνει αρχικά την ανάκτηση μεταδεδομένων (όνομα, τύπος, μέγεθος, τελική διεύθυνση) και στη συνέχεια την ασύγχρονη λήψη και ανάρτηση στον αποθηκευτικό χώρο. Ο χρήστης ενημερώνεται σε πραγματικό χρόνο για την πορεία της μεταφοράς μέσω καναλιού WebSocket, γεγονός που καθιστά δυνατή την ενσωμάτωση ακόμη και μεγάλων αρχείων χωρίς χειροκίνητη διαχείριση ή επαν-ανέβασμα.
Καθ’ όλη τη διαδικασία εφαρμόζονται αυστηροί έλεγχοι πρόσβασης και εγκυρότητας. Το σύστημα επαληθεύει ότι ο χρήστης διαθέτει δικαίωμα θέασης ή μεταβολής της υπόθεσης σύμφωνα με το μοντέλο RBAC και τις πολιτικές ορατότητας (private, local, public). Επιπλέον, ελέγχονται όρια μεγέθους και άλλες παράμετροι εγκυρότητας, ενώ σε κάθε περίπτωση ο χρήστης λαμβάνει σαφή μηνύματα ανατροφοδότησης που μειώνουν την αβεβαιότητα. Η καταχώριση είναι ανθεκτική σε επαναλήψεις: αν επιχειρηθεί ξανά η μεταφόρτωση ίδιου αρχείου στην ίδια υπόθεση, ενημερώνεται η υπάρχουσα εγγραφή αντί να δημιουργηθεί διπλότυπο, διατηρώντας έτσι καθαρό το ιστορικό της υπόθεσης (Case).
Με την ολοκλήρωση της μεταφόρτωσης και την καταχώριση των μεταδεδομένων, ενεργοποιείται αυτόματα η ροή επεξεργασίας. Τα Document περνούν από στάδια parsing και εξαγωγής κειμένου, αναγνώρισης οντοτήτων (NER), δημιουργίας embeddings και ένταξης στους μηχανισμούς αναζήτησης και στο γράφο γνώσης. Ο χρήστης επιστρέφει στην προβολή της υπόθεσης, όπου η εφαρμογή

αποτυπώνει σε πραγματικό χρόνο την εξέλιξη της επεξεργασίας για κάθε έγγραφο. Η κατάσταση ενημερώνεται δυναμικά (από «uploaded» σε «processing», «parsed» κ.ο.κ.), μαζί με βασικά μεταδεδομένα όπως ο χρόνος αποστολής, η ανίχνευση γλώσσας, ο αριθμός των αποσπασμάτων που δημιουργήθηκαν για αναζήτηση και ο δείκτης ολοκλήρωσης των επιμέρους σταδίων. Η ζωντανή αυτή παρακολούθηση μειώνει την αβεβαιότητα, επιτρέπει την έγκαιρη διάγνωση προβλημάτων και ενισχύει την ιχνηλασιμότητα, καθώς ο αναλυτής βλέπει πότε ακριβώς κάθε στάδιο ολοκληρώθηκε και πότε το έγγραφο καθίσταται διαθέσιμο για αναζήτηση, εξαγωγή οντοτήτων και περαιτέρω ανάλυση.



Σχήμα 5.3: Προβολή υπόθεσης με ζωντανή ενημέρωση κατάστασης εγγράφων. Η διεπαφή αποτυπώνει σε πραγματικό χρόνο την πρόοδο των σταδίων (μεταφόρτωση, ανάλυση, αναγνώριση οντοτήτων, ενσωμάτωση στην αναζήτηση).

5.5 Σημασιολογική Εμπλουτισμένη Πληροφορία

5.5.1 Διαδραστική Επεξεργασία και Επαλήθευση Οντοτήτων (NER Workflow)

Η διαδικασία αναγνώρισης και επαλήθευσης οντοτήτων (Named Entity Recognition – NER) αποτελεί κρίσιμο βήμα στην αλυσίδα μετατροπής ενός εγγράφου σε σημασιολογικά εμπλουτισμένη πληροφορία. Μετά το parsing και την αρχική ανάλυση από το μοντέλο spaCy, οι οντότητες φορτώνονται στη διεπαφή NER Editor, όπου ο χρήστης μπορεί να τις εξετάσει, να τις διορθώσει ή να τις επεκτείνει.

Η διεπαφή οργανώνεται σε ξεχωριστές καρτέλες που υποστηρίζουν διαφορετικές προβολές: Model Extracted με τις αυτόματα ανιχνευμένες οντότητες, User Approved με τις οντότητες που έχει εγκρίνει ή τροποποιήσει ο χρήστης, και Saved Entities με τον πίνακα των καταχωρισμένων στη βάση δεδομένων. Η δομή αυτή επιτρέπει την άμεση σύγκριση ανάμεσα στην πρόβλεψη του μοντέλου και στην τελική εκδοχή που υιοθετείται για ανάλυση και αποθήκευση.

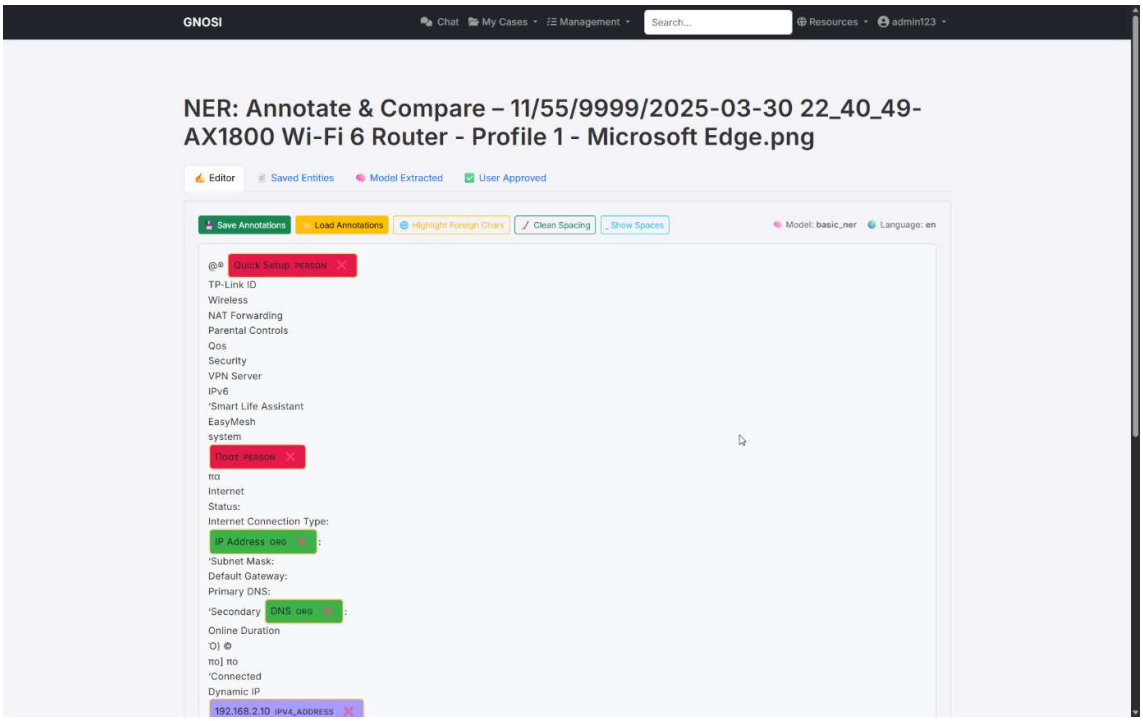
Στην καρτέλα Editor παρέχεται διαδραστικό περιβάλλον σήμανσης. Ο χρήστης μπορεί να επιλέξει αποσπάσματα κειμένου και να τα χαρακτηρίσει με labels, όπως PERSON, ORG ή LOC,

χρησιμοποιώντας έγχρωμες επισημάνσεις για εύκολη διάκριση. Παράλληλα, διαθέτει εργαλεία βελτίωσης του κειμένου:

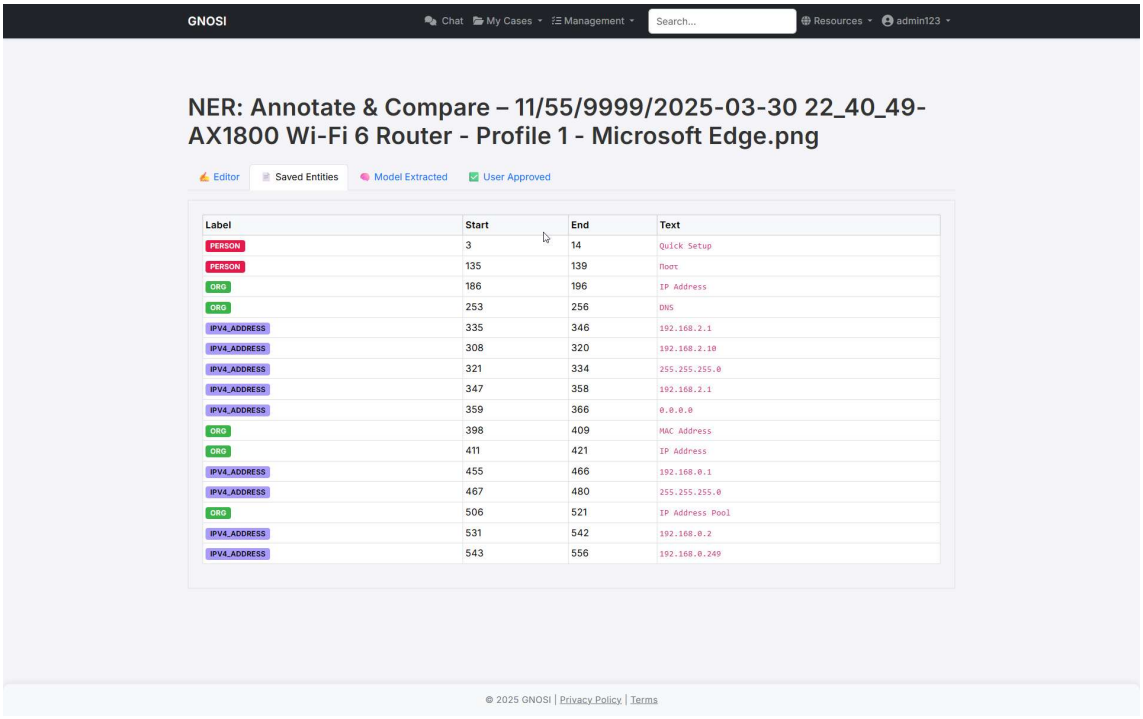
- επισήμανση «ξένων χαρακτήρων» (Greek/Latin),
- καθαρισμό πολλαπλών κενών,
- οπτικοποίηση κενών διαστημάτων,
- μαζική αφαίρεση επισημάνσεων.

Οι οντότητες μπορούν να αποθηκευτούν στη βάση δεδομένων μέσω της επιλογής Save, οπότε ενημερώνονται οι σχετικοί πίνακες (NEREntity, Document) και ενεργοποιείται η διαδικασία συγχρονισμού με το γράφημα γνώσης στο Neo4j. Με την επιλογή Load ανακτώνται οι ήδη καταχωρισμένες οντότητες, διασφαλίζοντας ανθεκτικότητα σε επανεκκινήσεις και συνέχιση της επεξεργασίας.

Η αλληλουχία «Model Extracted → User Approved → Saved Entities» καθιστά τη διαδικασία ελέγχου διαφανή και αναπαραγώγιμη, ενώ η αυτόματη ενημέρωση του γραφήματος γνώσης ενισχύει την ιχνηλασιμότητα σε επίπεδο υπόθεσης. Ο χρήστης έχει έτσι στη διάθεσή του ένα ολοκληρωμένο πλαίσιο διαχείρισης και επιβεβαίωσης οντοτήτων, που συνδυάζει αυτοματοποίηση και ανθρώπινη εποπτεία, εξασφαλίζοντας ισορροπία ανάμεσα στην αποδοτικότητα και την ακρίβεια.



Σχήμα 5.4: Διεπαφή NER Editor με εργαλεία επισημείωσης.



Σχήμα 5.5: Πίνακας Saved Entities με χρωματική κωδικοποίηση ανά τύπο οντότητας.

5.5.2 Αναφορές και Απεικόνιση με Γράφους

Η τελική αξιοποίηση των αναγνωρισμένων και εμπλουτισμένων δεδομένων πραγματοποιείται μέσα από τη δημιουργία αναφορών και την οπτικοποίησή τους σε μορφή γράφων. Οι αναφορές αυτές μπορεί να είναι είτε περιγραφικές, παρουσιάζοντας συνοπτικά στατιστικά στοιχεία για το περιεχόμενο (όπως αριθμός οντοτήτων και κατηγορίες), είτε αναλυτικές, εστιάζοντας στις σχέσεις και τις συσχετίσεις μεταξύ εγγράφων, οντοτήτων και υποθέσεων.

Το δεύτερο είδος αναφορών είναι και το πιο ισχυρό, καθώς επιτρέπει την αναγνώριση κοινών στοιχείων ανάμεσα σε διαφορετικά τεκμήρια και υποθέσεις, αναδεικνύοντας πιθανά μοτίβα, κυκλώματα ή επαναλαμβανόμενες συμπεριφορές.

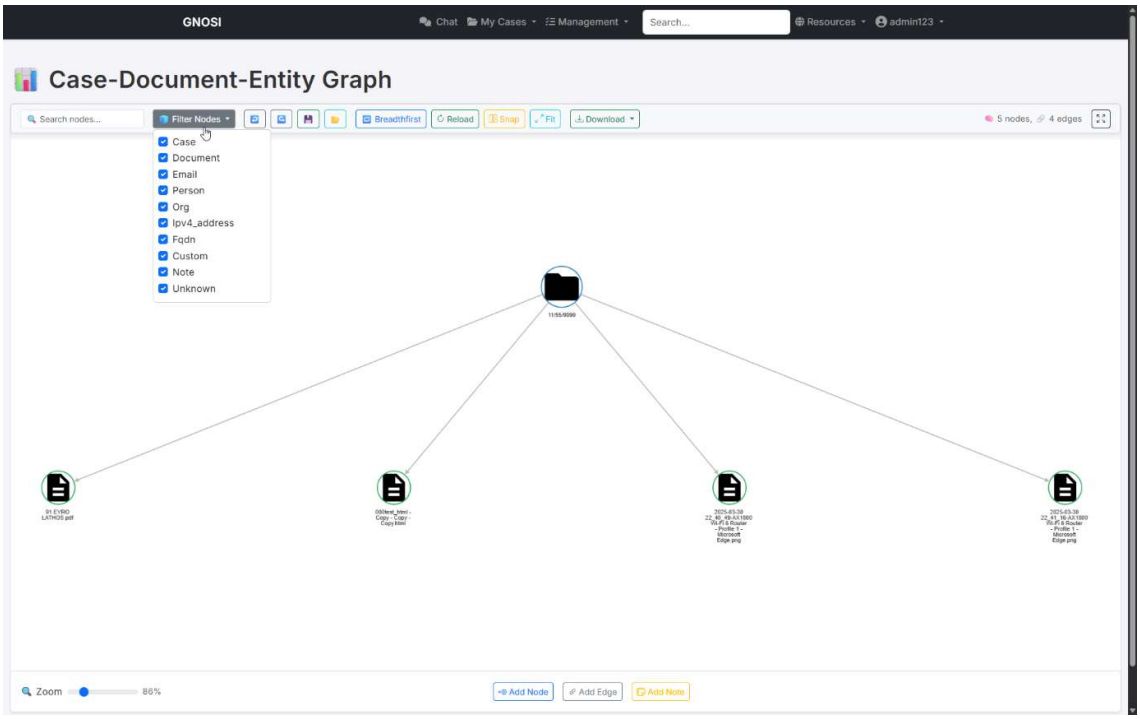
Η οπτικοποίηση με γράφους αποτελεί ένα διαδραστικό εργαλείο που δίνει στον χρήστη τη δυνατότητα να εξερευνήσει τις συσχετίσεις με άμεσο και ευανάγνωστο τρόπο. Κάθε υπόθεση (Case) εμφανίζεται ως κόμβος που περιέχει τα σχετικά έγγραφα (Documents), ενώ οι οντότητες (Entities) που αναγνωρίζονται από τη διαδικασία επεξεργασίας κειμένου συνδέονται με τα έγγραφα από τα οποία προέκυψαν. Η απεικόνιση αυτή επιτρέπει στον χρήστη να ανακαλύψει συνδέσεις που δεν θα ήταν εύκολο να εντοπιστούν μόνο με την ανάγνωση των δεδομένων.

Ο χρήστης μπορεί να πλοηγηθεί ελεύθερα στον γράφο, να αλλάξει τη διάταξή του ώστε να αναδεικνύει διαφορετικές όψεις, να αναζητήσει συγκεκριμένους κόμβους ή να εφαρμόσει φίλτρα που περιορίζουν την απεικόνιση σε συγκεκριμένες κατηγορίες, όπως emails ή IPs. Παράλληλα, έχει τη δυνατότητα να επεκτείνει δυναμικά έναν κόμβο για να δει τις συνδεόμενες πληροφορίες, να προσθέσει σημειώσεις ή σχόλια, καθώς και να τροποποιήσει ή να επισημάνει τις σχέσεις μεταξύ κόμβων.

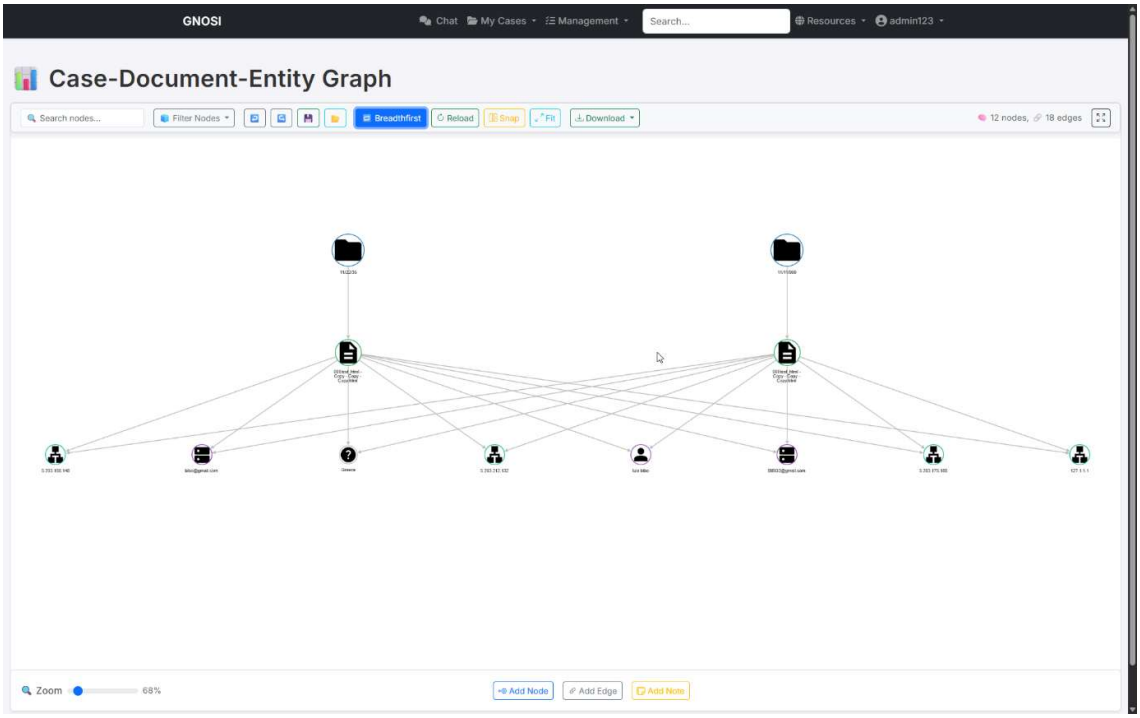
Η απεικόνιση δεν λειτουργεί μόνο ως στατική αναπαράσταση, αλλά ως εργαλείο διερεύνησης. Κάθε ενέργεια μπορεί να αναιρεθεί ή να αποθηκευτεί, ενώ ο γράφος μπορεί να εξαχθεί σε μορφή εικόνας ή

δεδομένων ώστε να αξιοποιηθεί σε αναφορές ή σε περαιτέρω ανάλυση. Μέσα από την αλληλεπίδραση αυτή, η απεικόνιση γίνεται ουσιαστικό μέρος της διαδικασίας ανάλυσης, προσφέροντας στον ερευνητή τη δυνατότητα να εντοπίζει κρίσιμες συνδέσεις, να αναγνωρίζει επαναλαμβανόμενα πρότυπα και να υποστηρίζει τα ευρήματά του με σαφή και κατανοητό τρόπο.

Η διαδραστική φύση του εργαλείου το καθιστά πολύτιμο όχι μόνο για την ανάλυση, αλλά και για την επικοινωνία των αποτελεσμάτων. Μέσω της άμεσης οπτικής αναπαράστασης, τα δεδομένα μετατρέπονται σε γνώση, ενώ η πολυπλοκότητα των σχέσεων απλοποιείται σε μια εικόνα που μπορεί να πείσει, να υποστηρίξει αποφάσεις και να τροφοδοτήσει περαιτέρω έρευνα.



Σχήμα 5.6: Γραφική αναπαράσταση μίας υπόθεσης και των εγγράφων που περιλαμβάνει.



Σχήμα 5.7: Γραφική αναπαράσταση μίας υπόθεσης, των εγγράφων και των οντοτήτων που περιλαμβάνει καθώς και κοινών οντοτήτων που παρουσιάζει με άλλη υπόθεση.

5.6 Αναζήτηση και Αλληλεπίδραση

5.6.1 Κλασική & Υβριδική Αναζήτηση

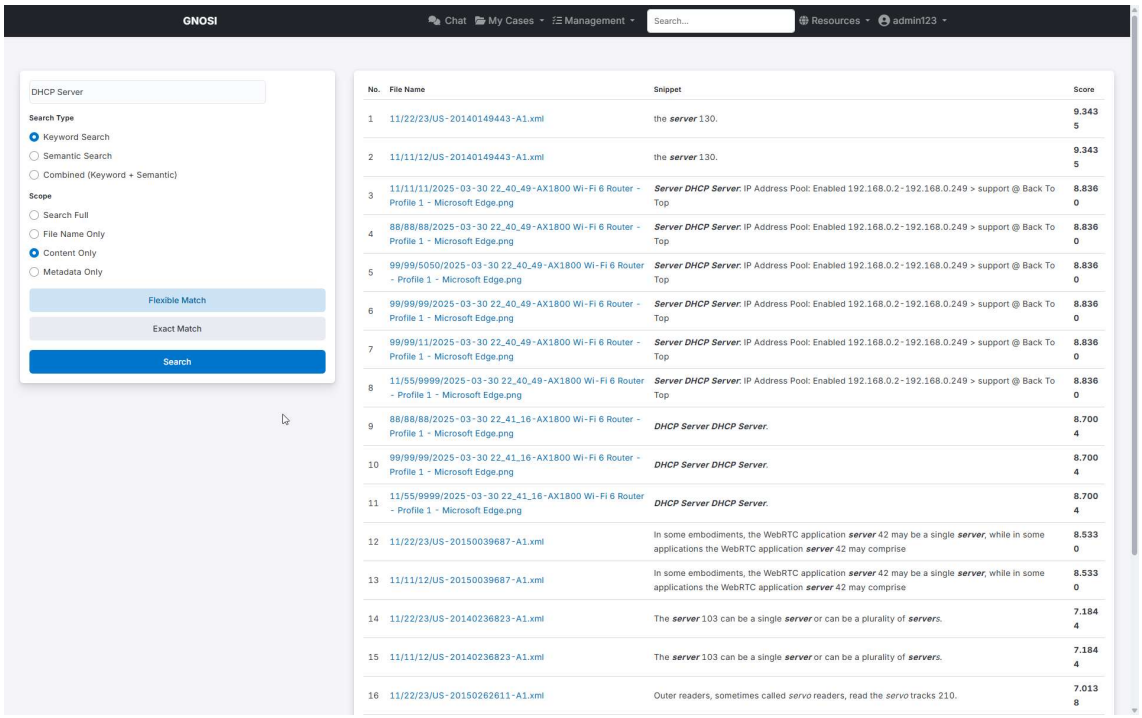
Η αναζήτηση αποτελεί το βασικό εργαλείο εντοπισμού πληροφορίας σε όλο το αποθετήριο. Ο χρήστης ξεκινά είτε από το πεδίο γρήγορης αναζήτησης στη γραμμή πλοήγησης είτε από την ειδική σελίδα, όπου συγκεντρώνονται όλα τα χειριστήρια. Το πεδίο αναζήτησης «θυμάται» τους τελευταίους όρους και τους προτείνει ως ιστορικό, διευκολύνοντας την επαναχρησιμοποίηση ερωτημάτων.

Η διεπαφή προσφέρει τρεις τρόπους αναζήτησης. Η αναζήτηση με λέξεις-κλειδιά εξυπηρετεί τα κλασικά σενάρια, με επιλογή ευέλικτης ή ακριβούς ταύτισης, ώστε ο χρήστης να επιλέγει αν δέχεται παραλλαγές ή απαιτεί ακριβές κείμενο. Η σημασιολογική αναζήτηση εντοπίζει κείμενα με παρόμοιο νόημα ακόμη κι όταν οι όροι δεν συμπίπτουν, ενώ ο συνδυαστικός τρόπος φέρνει ισορροπία ανάμεσα στην ακριβή φρασεολογία και στη νοηματική συνάφεια. Για τις αναζητήσεις με λέξεις-κλειδιά ο χρήστης καθορίζει και το «πεδίο δράσης»: σε ολόκληρο το τεκμήριο, μόνο στο όνομα αρχείου, μόνο στο περιεχόμενο ή αποκλειστικά στα μεταδεδομένα. Όταν επιλέγονται μεταδεδομένα, η διεπαφή προτείνει διαθέσιμα κλειδιά, ώστε το φιλτράρισμα να γίνεται στοχευμένα.

Τα αποτελέσματα εμφανίζονται σε ευανάγνωστο πίνακα, με αρίθμηση και πληροφορία εύρους (π.χ. «εμφανίζονται τα 21–40 από Χ»), ώστε να γίνεται σαφές το σημείο της πλοήγησης. Κάθε γραμμή παρουσιάζει το όνομα αρχείου ως σύνδεσμο προς την προβολή περιεχομένου, και όπου είναι διαθέσιμο εμφανίζεται αντιπροσωπευτικό απόσπασμα (snippet) με έντονη επισήμανση των όρων. Στις σημασιολογικές αναζητήσεις συνυπάρχει και μια ενδεικτική βαθμολογία συνάφειας, που βοηθά να

ξεχωρίζουν τα πιο σχετικά ευρήματα. Η σελιδοποίηση είναι ενσωματωμένη και επιτρέπει γρήγορη μετακίνηση στα επόμενα σύνολα χωρίς απώλεια του ερωτήματος.

Για τις σημασιολογικές αναζητήσεις ο χρήστης ρυθμίζει το πλήθος των κορυφαίων αποτελεσμάτων μέσα από ένα απλό χειριστήριο, πετυχαίνοντας ισορροπία ανάμεσα στο βάθος της αναζήτησης και την ταχύτητα. Σε κάθε αλληλεπίδραση η διεπαφή δίνει σαφή οπτική ανατροφοδότηση (ένδειξη φόρτωσης, μετρητές, επισημάνσεις όρων), ώστε να μειώνεται η αβεβαιότητα και να διατηρείται η αίσθηση ελέγχου. Συνολικά, η εμπειρία συνδυάζει την απλότητα της κλασικής αναζήτησης με την ισχύ της σημασιολογικής ανάκτησης. Ο χρήστης ξεκινά γρήγορα, φιλτράρει με ακρίβεια, βλέπει αμέσως τα πιο σχετικά αποσπάσματα και μεταβαίνει στο πλήρες τεκμήριο με ένα κλικ — μια ροή που υποστηρίζει τόσο την καθημερινή χρήση όσο και απαιτητικά ερευνητικά σενάρια.



Σχήμα 5.8: Οθόνη αναζήτησης με επιλογή τρόπου (λέξεις-κλειδιά, σημασιολογική, συνδυαστική), ρύθμιση πεδίου/ταιριάσματος και αποτελέσματα με επισημασμένα αποσπάσματα και σελιδοποίηση.

5.6.2 Συνομιλιακή Αναζήτηση (RAG Bot)

Η συνομιλιακή διεπαφή ενσωματώνει τη λογική της αναζήτησης με ανάκτηση και παραγωγή (Retrieval-Augmented Generation – RAG), προσφέροντας στον χρήστη ένα δυναμικό περιβάλλον αλληλεπίδρασης με τα δεδομένα του συστήματος. Δεν περιορίζεται σε παραδοσιακές λειτουργίες αναζήτησης, αλλά επιτρέπει τη διατύπωση ερωτημάτων σε φυσική γλώσσα και την άμεση παραγωγή απαντήσεων, οι οποίες συνοδεύονται από αποσπάσματα και παραπομπές σε σχετικά έγγραφα, διασφαλίζοντας έτσι τη διαφάνεια και την τεκμηρίωση.

Στο ανώτερο τμήμα της διεπαφής παρατίθενται ρυθμίσεις που καθορίζουν τη συμπεριφορά του μοντέλου, όπως η επιλογή αλγορίθμου, οι παράμετροι δημιουργικότητας (θερμοκρασία) και ο αριθμός των ανακτηθέντων τεκμηρίων (Top-K). Παράλληλα, ενσωματώνεται χρονοδιακόπτης συνεδρίας για την ορθολογική διαχείριση των πόρων και την έγκαιρη ενημέρωση του χρήστη σχετικά με τη διάρκεια

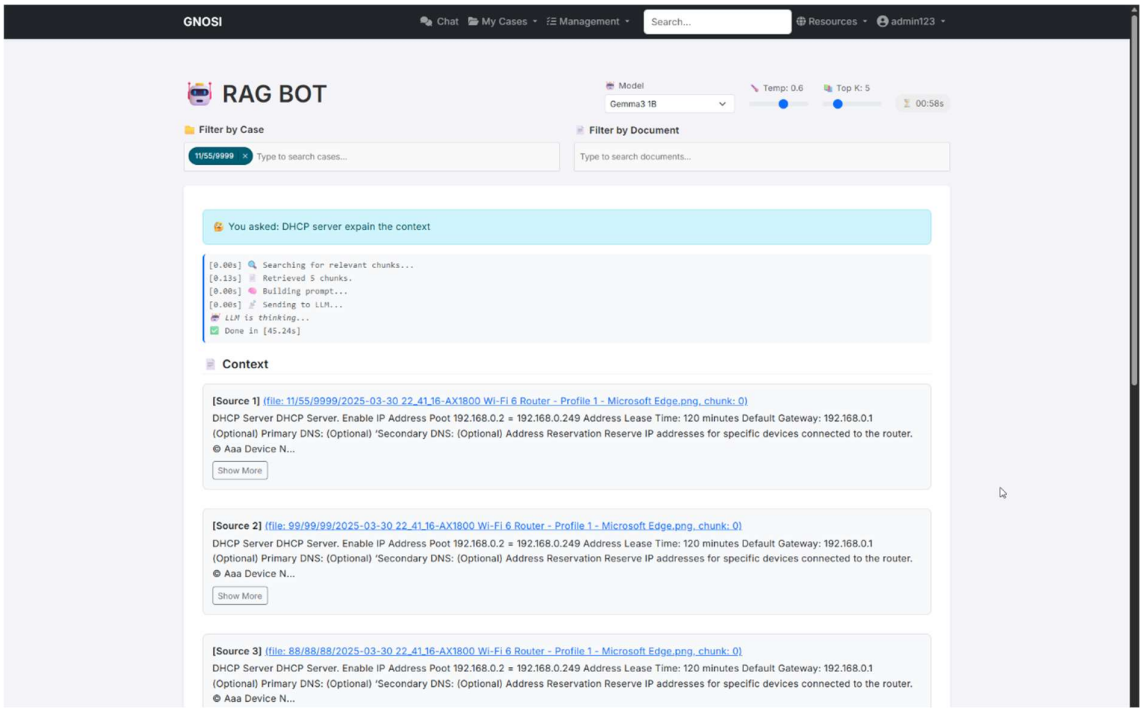
ισχύος της συνεδρίας του. Επιπλέον, φίλτρα περιορισμού της αναζήτησης σε συγκεκριμένες υποθέσεις ή έγγραφα διευκολύνουν τη στοχευμένη διερεύνηση.

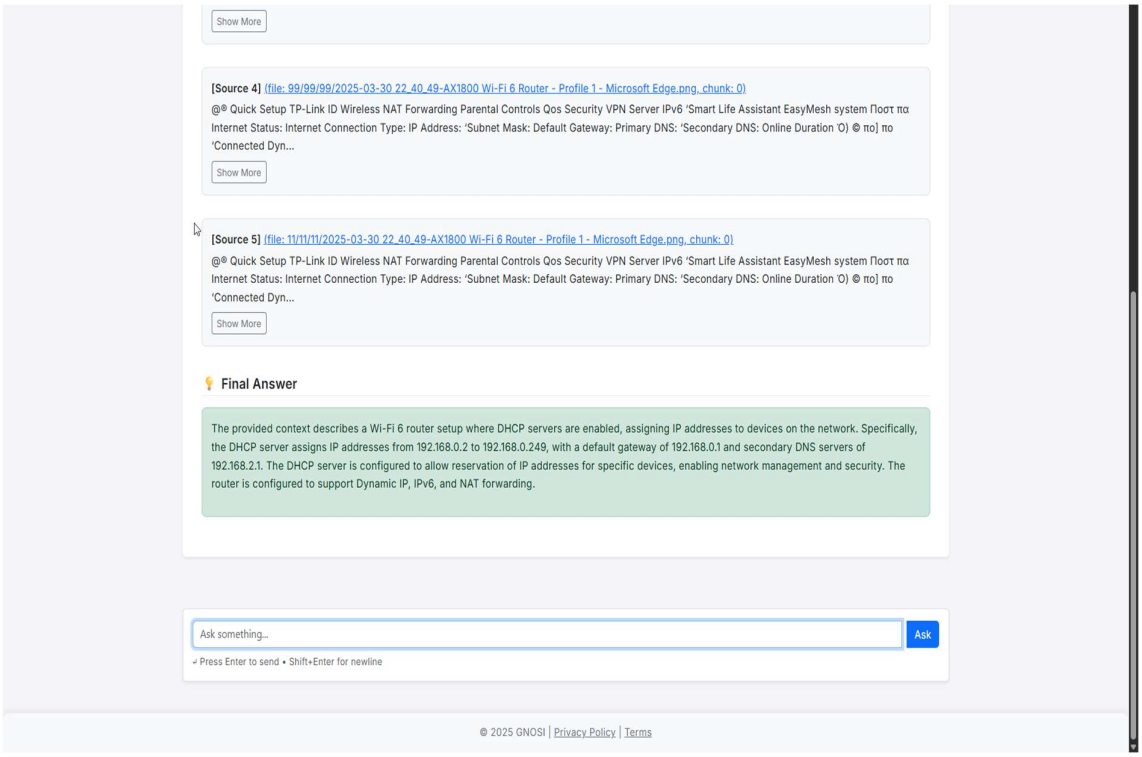
Κάθε αλληλεπίδραση οργανώνεται σε αυτόνομες ενότητες («κάρτες»), όπου αποτυπώνονται το αρχικό ερώτημα του χρήστη, τα ενδιάμεσα βήματα της διαδικασίας αναζήτησης, το σύνολο των πηγών που αξιοποιήθηκαν (με τίτλο, σύνδεσμο και αντιπροσωπευτικό απόσπασμα), καθώς και η παραγόμενη απάντηση σε φυσική γλώσσα. Με τον τρόπο αυτό η τελική απόκριση δεν εμφανίζεται ως «μαύρο κουτί», αλλά συνοδεύεται από σαφείς παραπομπές και επεξηγήσεις, ενισχύοντας την αξιοπιστία της.

Ιδιαίτερη έμφαση δίνεται στην εμπειρία χρήσης. Η καταχώριση ερωτημάτων είναι απλή, με διακριτή διάκριση ανάμεσα σε νέα γραμμή και υποβολή, ενώ οι ρυθμίσεις και τα επιλεγμένα φίλτρα αποθηκεύονται τοπικά, ώστε να επαναχρησιμοποιούνται σε μελλοντικές συνεδρίες. Σε περιπτώσεις υπέρβασης του ορίου ταυτόχρονων συνεδριών, το σύστημα ενημερώνει με σαφήνεια και επιχειρεί αυτόματη επανασύνδεση, περιορίζοντας την πιθανότητα διακοπής της αναλυτικής εργασίας. Πριν από την πρώτη χρήση, ο χρήστης λαμβάνει προειδοποίηση σχετικά με τα όρια της τεχνολογίας, καθώς οι παραγόμενες απαντήσεις, αν και τεκμηριωμένες, απαιτούν πάντοτε κριτική αποτίμηση και διασταύρωση με έγκυρες πηγές.

Πέραν της γενικής αναζήτησης, η συνομιλιακή διεπαφή υποστηρίζει και εξειδικευμένη αλληλεπίδραση σε επίπεδο υπόθεσης ή εγγράφου. Μέσω φίλτρων, ο χρήστης μπορεί να περιορίσει τη συζήτηση σε συγκεκριμένο φάκελο (case) ή σε επιλεγμένα τεκμήρια (documents), ώστε οι απαντήσεις να εδράζονται αποκλειστικά στο περιεχόμενό τους. Η λειτουργικότητα αυτή είναι ιδιαίτερα χρήσιμη, καθώς καθιστά δυνατή την εις βάθος μελέτη των δεδομένων μιας μεμονωμένης υπόθεσης, την εξαγωγή απαντήσεων βασισμένων σε συγκεκριμένα έγγραφα και τη συγκριτική ανάλυση τεκμηρίων εντός της ίδιας υπόθεσης, με διατήρηση της συνομιλιακής συνέχειας.

Συνολικά, ο RAG Bot δεν λειτουργεί απλώς ως μηχανή αναζήτησης, αλλά ως βοηθός ανάλυσης με επίγνωση συμφραζομένων, επιτρέποντας στον χρήστη να προσαρμόζει το εύρος της διερεύνησης ανάλογα με τις ανάγκες του. Έτσι, η αναζήτηση μετατρέπεται σε μια διαδραστική και τεκμηριωμένη διαδικασία, που ενισχύει την ερευνητική εμβάθυνση και τη λήψη αποφάσεων.



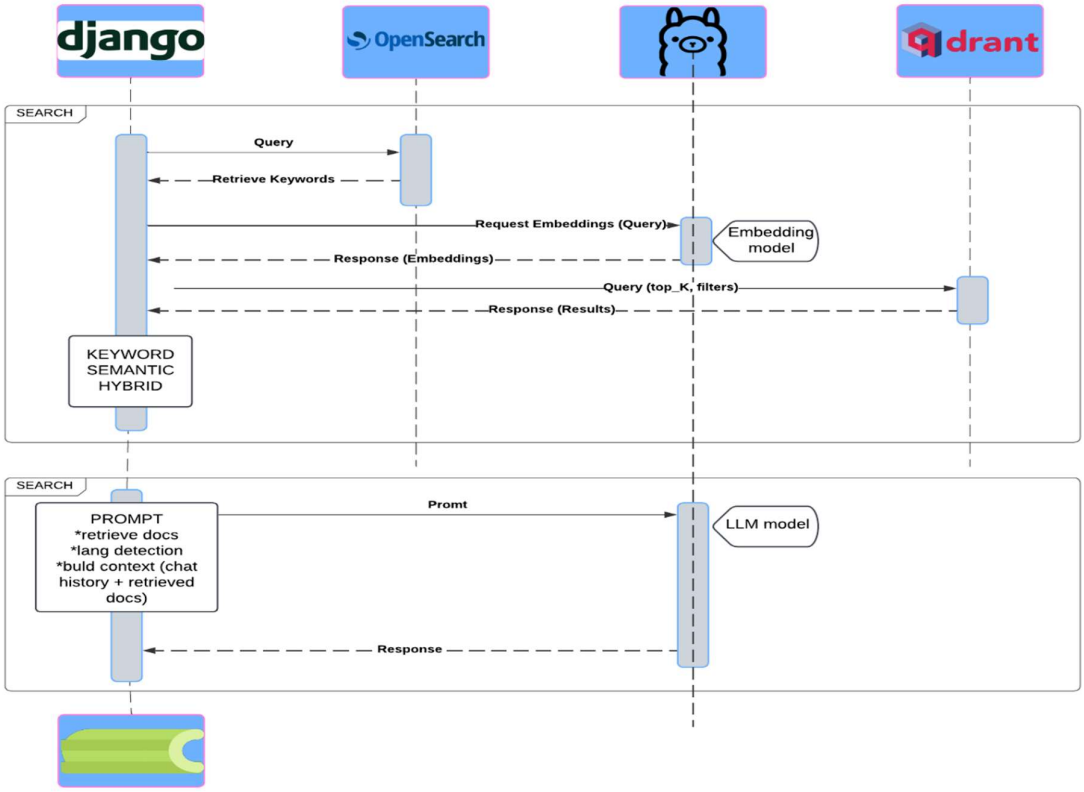


Σχήμα 5.9: Οθόνη υποβολής ερωτημάτων στο RAG Bot.

5.7 Αναλυτική Ροή Υβριδικής Αναζήτησης και LLM Επεξεργασίας

Οι δυνατότητες αναζήτησης και συνομιλιακής διεπαφής που παρουσιάστηκαν στις προηγούμενες ενότητες βασίζονται σε μια πολυεπίπεδη ροή επεξεργασίας, η οποία συνδυάζει κλασικές τεχνικές αναζήτησης με λέξεις-κλειδιά, σημασιολογική ανάκτηση μέσω διανυσματικών αναπαραστάσεων, και τελικά την αξιοποίηση μεγάλων γλωσσικών μοντέλων (LLMs) για τη σύνθεση απαντήσεων σε φυσική γλώσσα. Η ροή αυτή επιτρέπει αφενός τον ακριβή εντοπισμό πληροφοριών, αφετέρου την ερμηνεία και οργάνωσή τους σε τεκμηριωμένες απαντήσεις, αναβαθμίζοντας τη διαδικασία αναζήτησης σε ένα ολοκληρωμένο εργαλείο ανάλυσης.

Στην ενότητα που ακολουθεί περιγράφονται αναλυτικά τα στάδια της υβριδικής αναζήτησης και της αλληλεπίδρασής της με τα LLMs, από την υποβολή του αρχικού ερωτήματος έως την παρουσίαση της τελικής απόκρισης στον χρήστη.



Σχήμα 5.10: Διάγραμμα Ροής Υβριδικής Αναζήτησης και Επεξεργασίας Prompts μέσω LLM

Αναλυτική παρουσίαση των βημάτων έχουν ως εξής:

User → Django

Ο χρήστης υποβάλλει ένα ερώτημα μέσω της διεπαφής (UI). Το query μπορεί να είναι απλή αναζήτηση με λέξεις-κλειδιά ή πιο σύνθετη ερώτηση που απαιτεί παραγωγή απάντησης.

Django → OpenSearch

Το query αποστέλλεται στο υποσύστημα **OpenSearchService**, το οποίο εκτελεί αναζήτηση με βάση λέξεις-κλειδιά και επιστρέφει αποτελέσματα exact/partial match.

Django → EmbeddingService (Ollama / Sentence Transformers)

Παράλληλα, το ίδιο query μετατρέπεται σε διανυσματική αναπαράσταση (embedding). Το embedding μπορεί να παραχθεί είτε μέσω Sentence Transformers είτε μέσω LLM μοντέλου που τρέχει τοπικά μέσω Ollama.

EmbeddingService → Django

Το παραγόμενο vector επιστρέφεται στο backend, έτοιμο για χρήση σε semantic search.

Django → Qdrant

Το vector αποστέλλεται στην υπηρεσία **QdrantVectorStorageService** για αναζήτηση εγγράφων με βάση τη σημασιολογική τους συνάφεια και το Qdrant επιστρέφει τα πιο σχετικά έγγραφα βάσει cosine similarity.

Qdrant → Django

Τα semantic αποτελέσματα επιστρέφονται στο backend.

OpenSearch → Django

Τα keyword-based αποτελέσματα από το OpenSearch συγχωνεύονται με τα semantic results για παραγωγή υβριδικού συνόλου.

Django → UI

Η διεπαφή εμφανίζει τα υβριδικά αποτελέσματα, παρέχοντας ένδειξη για το ποια προέρχονται από keyword match και ποια από semantic match.

User Prompt → Django (Prompt Builder)

Ο χρήστης μπορεί να επιλέξει έγγραφα και να υποβάλει νέο prompt για παραγωγή απάντησης. Το σύστημα δημιουργεί ένα structured prompt που περιλαμβάνει:

- Επιλεγμένα context documents
- Ιστορικό συνομιλίας
- Εντοπισμό γλώσσας (lang detection)

Django → Ollama LLM Service

Το prompt αποστέλλεται σε LLM που εκτελείται τοπικά μέσω Ollama (π.χ. Gemma, Mistral, LLaMA). Το μοντέλο επεξεργάζεται το prompt, αξιοποιεί τα context documents και παράγει απάντηση σε φυσική γλώσσα.

Ollama LLM Service → Django

Η απάντηση επιστρέφεται στο backend.

Django → UI

Η τελική απάντηση παρουσιάζεται στον χρήστη είτε ως μήνυμα συνομιλίας (chat mode) είτε ως structured report.

5.8 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε η λειτουργική ροή της εφαρμογής από την πλευρά του χρήστη. Αναλύθηκαν οι διαδικασίες σύνδεσης και ασφάλειας, η δημιουργία υποθέσεων, η μεταφόρτωση και επεξεργασία εγγράφων, η αναγνώριση και επαλήθευση οντοτήτων, καθώς και η οπτικοποίηση μέσω γράφων. Στη συνέχεια, περιγράφηκαν οι μηχανισμοί υβριδικής αναζήτησης και η ενσωμάτωση LLM για παραγωγή απαντήσεων σε φυσική γλώσσα.

Μέσα από αυτήν την ολοκληρωμένη ροή γίνεται φανερό πώς το σύστημα μετατρέπει αδόμητα δεδομένα σε αξιοποιήσιμη γνώση, προσφέροντας ένα περιβάλλον που συνδυάζει την παραδοσιακή αναζήτηση με τη δύναμη της τεχνητής νοημοσύνης. Το επόμενο κεφάλαιο εστιάζει στην τεχνική αρχιτεκτονική και τις υποδομές που καθιστούν δυνατές αυτές τις λειτουργίες, προσφέροντας πιο αναλυτική κατανόηση των επιμέρους τεχνολογιών.

Κεφάλαιο 6ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

6.1 Συμπεράσματα Υλοποίησης

Η υλοποίηση της εφαρμογής απέδειξε ότι είναι εφικτή η ανάπτυξη ενός ολοκληρωμένου συστήματος επεξεργασίας και ανάλυσης αδόμητων δεδομένων με τη χρήση σύγχρονων τεχνολογιών ανοιχτού κώδικα. Η αρχιτεκτονική που σχεδιάστηκε συνδύασε επιτυχώς τεχνολογίες αναγνώρισης οντοτήτων (spaCy), μηχανές αναζήτησης και ευρετηρίασης (OpenSearch, Qdrant), βάσεις γραφημάτων (Neo4j), καθώς και μηχανισμούς ασύγχρονης επεξεργασίας (Celery/Redis), δημιουργώντας μια ευέλικτη και επεκτάσιμη υποδομή.

Η χρήση μοντέλου RBAC (Role-Based Access Control) και σύγχρονων μηχανισμών ασφάλειας (2FA, HTTPS, κρυπτογράφηση) ενίσχυσε την αξιοπιστία του συστήματος, διασφαλίζοντας ότι τα δεδομένα παραμένουν προστατευμένα σε όλα τα στάδια. Παράλληλα, η υποστήριξη υβριδικής αναζήτησης (λέξεις-κλειδιά, σημασιολογική και συνδυαστική) σε συνδυασμό με το υποσύστημα RAG Bot κατέστησε την εφαρμογή ιδιαίτερα φιλική και αποδοτική για τον τελικό χρήστη.

Συνολικά, η εργασία ανέδειξε τη δυνατότητα δημιουργίας ενός αυτοματοποιημένου, ασφαλούς και επεκτάσιμου εργαλείου που ανταποκρίνεται σε πραγματικές ανάγκες οργανισμών, μειώνοντας το κόστος και τον χρόνο διαχείρισης δεδομένων.

6.2 Πλεονεκτήματα της Αρχιτεκτονικής

Η προτεινόμενη λύση διαθέτει ορισμένα σημαντικά πλεονεκτήματα:

- **Επεκτασιμότητα και ευελιξία**: Η χρήση microservices και containerization επιτρέπει εύκολη κλιμάκωση και προσαρμογή σε διαφορετικά περιβάλλοντα.
- **Ασφάλεια**: Ενσωμάτωση μηχανισμών ταυτοποίησης, κατακερματισμού κωδικών και κρυπτογράφησης επικοινωνίας.
- **Υποστήριξη ετερογενών δεδομένων**: Δυνατότητα επεξεργασίας εγγράφων διαφόρων μορφών (PDF, εικόνες, κείμενα).
- **Αποτελεσματική αναζήτηση**: Συνδυασμός λεξικής, σημασιολογικής και υβριδικής αναζήτησης με δυνατότητα συσχέτισης δεδομένων.
- **Οπτικοποίηση σχέσεων**: Η χρήση Neo4j επιτρέπει την αποτύπωση σύνθετων σχέσεων με τρόπο εύληπτο.
- **Αυτονομία**: Η χρήση τοπικών LLMs μέσω OLLAMA εξασφαλίζει ανεξαρτησία από εξωτερικούς παρόχους.
- **Ευχρηστία**: Φιλικό γραφικό περιβάλλον που καθιστά δυνατή τη χρήση της εφαρμογής και από μη τεχνικούς χρήστες.

6.3 Προτάσεις Βελτίωσης

6.3.1 Ακρίβεια στην Αναγνώριση Οντοτήτων (NER)

Παρότι η υλοποίηση με spaCy προσφέρει ήδη ικανοποιητικά αποτελέσματα στην αναγνώριση οντοτήτων σε ελληνικά και αγγλικά κείμενα, η ακρίβεια μπορεί να βελτιωθεί περαιτέρω μέσα από την εκπαίδευση εξειδικευμένων μοντέλων σε **domain-specific δεδομένα**. Η χρήση των datasets που έχουν ήδη συγκεντρωθεί στο πλαίσιο της εφαρμογής (π.χ. έγγραφα νομικού, επιχειρησιακού ή τεχνολογικού

περιεχομένου) μπορεί να λειτουργήσει ως βάση για fine-tuning, ενισχύοντας την ικανότητα του συστήματος να αναγνωρίζει όρους, ονόματα και οντότητες που είναι κρίσιμες για τον συγκεκριμένο τομέα. Με αυτόν τον τρόπο, το NER θα αποκτήσει μεγαλύτερη ακρίβεια και εξειδίκευση, ελαχιστοποιώντας τα λάθη και αυξάνοντας τη χρησιμότητα των αποτελεσμάτων για τους τελικούς χρήστες.

6.3.2 Βελτίωση της Συνομιλιακής Αναζήτησης (RAG Bot)

Το υπάρχον υποσύστημα **συνομιλιακής αναζήτησης** (RAG Bot) μπορεί να εξελιχθεί περαιτέρω με την αξιοποίηση πιο ισχυρών γλωσσικών μοντέλων (LLMs), τα οποία θα λειτουργούν σε υποδομές υψηλών επιδόσεων με υποστήριξη GPU. Με αυτόν τον τρόπο, το σύστημα θα μπορεί να επεξεργάζεται μεγαλύτερους όγκους δεδομένων, να δίνει ταχύτερες και πιο ακριβείς απαντήσεις και να χειρίζεται πιο σύνθετα ερωτήματα. Η ενίσχυση της υπολογιστικής ισχύος θα βελτιώσει τη συνολική εμπειρία του χρήστη, ενώ η χρήση βελτιστοποιημένων LLMs για το συγκεκριμένο domain θα συμβάλει στην παραγωγή πιο αξιόπιστων και εξειδικευμένων αποτελεσμάτων. Έτσι, το RAG Bot θα μετατραπεί σε έναν προηγμένο «έξυπνο βοηθό» που μπορεί να ανταποκρίνεται αποτελεσματικά στις αυξανόμενες απαιτήσεις ανάλυσης και πληροφόρησης.

6.3.3 Καθορισμός σύνθετων ερωτημάτων στη Neo4j και παρουσίαση στη διεπαφή γράφων

Μια σημαντική βελτίωση θα ήταν η υποστήριξη **σύνθετων ερωτημάτων στη Neo4j** με χρήση παραμετρικών φίλτρων (όπως χρονικά διαστήματα, τύποι οντοτήτων ή επίπεδο βάθους αναζήτησης). Τα αποτελέσματα αυτών των ερωτημάτων θα εμφανίζονται απευθείας στη **διαδραστική διεπαφή γράφων**, επιτρέποντας στους χρήστες να εξερευνούν σχέσεις και συσχετίσεις με μεγαλύτερη ευκολία και σαφήνεια. Με αυτόν τον τρόπο, το σύστημα θα παρέχει πιο ευέλικτη και στοχευμένη ανάλυση, βελτιώνοντας την εμπειρία και την αποτελεσματικότητα του τελικού χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Online Documentation / Web Resources

[1] Django Software Foundation, “Django Web Framework,” *Django Project*, 2024. [Online]. Available: https://www.djangoproject.com/

[2] FastAPI, “FastAPI Web Framework,” *FastAPI Docs*, 2024. [Online]. Available: https://fastapi.tiangolo.com/

[3] Celery Project, “Celery Distributed Task Queue,” *Celery Docs*, 2024. [Online]. Available: https://docs.celeryq.dev/

[4] MinIO Inc., “MinIO: Object Storage for AI,” *MinIO Docs*, 2024. [Online]. Available: https://min.io/

[5] Apache Software Foundation, “Apache Tika,” *Apache Tika Project*, 2024. [Online]. Available: https://tika.apache.org/

[6] Explosion AI, “spaCy NLP Library,” *spaCy.io*, 2024. [Online]. Available: https://spacy.io/

[7] Sentence Transformers, “SBERT: Sentence Embeddings Using BERT,” *SBERT*, 2024. [Online]. Available: https://www.sbert.net/

[8] OpenSearch Project, “OpenSearch: Open Source Search Engine,” *OpenSearch.org*, 2024. [Online]. Available: https://opensearch.org/

[9] Qdrant, “Qdrant Vector Search Engine,” *Qdrant Docs*, 2024. [Online]. Available: https://qdrant.tech/

[10] Neo4j Inc., “Neo4j Graph Database,” *Neo4j.com*, 2024. [Online]. Available: https://neo4j.com/

[11] OLLAMA, “Run LLMs Locally with Ollama,” *Ollama*, 2024. [Online]. Available: https://ollama.com/

[12] Docker Inc., “Docker: Empowering App Development for Developers,” *Docker Docs*, 2024. [Online]. Available: https://www.docker.com/

[17] J. Lewis and M. Fowler, “Microservices: A definition of this new architectural term,” 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html

[26] S. Sayfullin, “Full-text search in Django with Elasticsearch,” *RealPython.com*, 2023. [Online]. Available: https://realpython.com/django-elasticsearch

[29] A. Solem, *Celery: Distributed Task Queue Documentation*, 2024. [Online]. Available: https://docs.celeryq.dev/

[37] *Django + Celery + Redis: Real world use cases*. *RealPython.com*. [Online]. Available: https://realpython.com/asynchronous-tasks-with-django/

[40] Amazon Web Services, “Amazon S3 API reference,” 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/API/

[42] *High availability in MinIO with erasure coding and distributed mode*. MinIO Blog, 2022. [Online]. Available: https://blog.min.io

[43] *Using django-storages with MinIO*. Django Project, 2024. [Online]. Available: https://django-storages.readthedocs.io

[45] *Why MinIO is AWS S3 compatible*. MinIO, 2024. [Online]. Available: https://min.io

[47] *Bucket notification guide*. MinIO Documentation, 2023. [Online]. Available: https://min.io/docs/minio/linux/monitoring/bucket-notifications.html

[52] OpenAI, *OCR Techniques and Use Cases*, Tech. Brief, 2023. [Online]. Available: https://openai.com/research/ocr

[53] *Qdrant. Qdrant Vector Database Documentation*. 2025. [Online]. Available: https://qdrant.tech/documentation

[54] *Neo4j Graph Database Platform*. 2025 [online]. Available: https://neo4j.com/developer/

[55] Explosion AI. *spaCy Documentation*. 2025 [online]. Available https://spacy.io/usage
[56] FastAPI. *FastAPI – The High-performance Web Framework for Building APIs with Python*. 2025 [online]. Available: https://fastapi.tiangolo.com .
[59] Ollama. *Ollama Documentation*. 2025 [online]. Available https://ollama.com/library .

Βιβλία

[14] J. Bass, *Introduction to System Architecture*, 2nd ed. Cham, Switzerland: Springer, 2022.
[15] P. Clements et al., *Documenting Software Architectures: Views and Beyond*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2010.
[16] C. Richardson, *Microservices Patterns: With Examples in Java*. Shelter Island, NY, USA: Manning, 2019.
[21] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, 2nd ed. Berkeley, CA, USA: Apress, 2009.
[22] T. Winter and A. Phillips, *Mastering Django: Core*, 4th ed. Birmingham, U.K.: Packt Publishing, 2022.
[24] A. Moore, *Web Development with Django and React*. Sebastopol, CA, USA: O'Reilly Media, 2021.
[27] J. Gormley and C. Tong, *Elasticsearch: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, 2015.
[30] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.
[32] D. Lacey, *Building Scalable Python Applications with Celery and Redis*. Birmingham, U.K.: Packt Publishing, 2020.
[33] S. Sanfilippo, *Redis: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, 2023.
[35] R. Somasundaram, *Mastering Celery with Django*. Birmingham, U.K.: Packt Publishing, 2022.
[39] T. Anderson, *Object Storage for Digital Transformation*. Sebastopol, CA, USA: O'Reilly Media, 2020.
[46] J. Pease, *Mastering AWS S3: A Comprehensive Guide*. Sebastopol, CA, USA: O'Reilly Media, 2020.
[48] C. Mattmann and J. Zitting, *Tika in Action*. Shelter Island, NY, USA: Manning Publications, 2011.
[50] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Boston, MA, USA: Pearson, 2023.
[51] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland: Springer, 2022.

Journal Papers

[13] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001.
[20] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, pp. 2–11, Mar. 2014.

[25] P. Kluyver et al., “Deploying Django in enterprise environments,” *J. Web Eng.*, vol. 19, no. 4, pp. 332–345, 2023.
[31] J. Hsu and M. Snyder, “Using Celery with Redis in modern Django applications,” *J. Open Source Softw. Eng.*, vol. 7, no. 2, pp. 55–61, 2023.
[34] B. Huang and S. Palit, “Real-time task management with Redis message brokers,” *Int. J. Cloud Comput.*, vol. 8, no. 2, pp. 101–115, 2020.
[38] S. Ghemawat, H. Gobioff, and S. T. Leung, “The Google file system,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.
[41] A. Arora and R. Jain, “Secure object storage systems: Encryption and access control,” *IEEE Cloud Comput.*, 2021.

Conference Papers

[18] G. Lakas, “A comparative study of software architecture styles for web systems,” in *Proc. 14th Int. Conf. Softw. Technol.*, 2019, pp. 105–113.
[19] M. Armbrust et al., “Above the clouds: A Berkeley view of cloud computing,” EECS Department, Univ. California, Berkeley, Tech. Rep., 2009.
[23] M. Basham, “Understanding Django’s MVT architecture,” in *Proc. Python Web Conf.*, 2021.
[28] L. Xu and Y. Liu, “Hybrid document retrieval using OpenSearch and machine learning,” in *Proc. IEEE Int. Conf. Big Data*, 2022, pp. 765–772.
[36] H. Xu and L. Wang, “Scalable asynchronous architectures for web applications: A Redis–Celery integration,” in *Proc. IEEE Cloud Comput. Conf.*, 2021.
[44] N. Huber and G. Rizos, “Self-hosted cloud architectures for privacy-aware storage,” in *Proc. IEEE EuroS&P*, 2021.
[49] R. Smith, “An overview of the Tesseract OCR engine,” in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, 2007, pp. 629–633.
[57] D. Nadeau και S. Sekine, “A survey of named entity recognition and classification,” *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007
[58] M. Akbik, T. Bergmann, και R. Vollgraf, “Pooled contextualized embeddings for named entity recognition,” στο *Proceedings of NAACL-HLT 2019*, pp. 724–728.
[60] Touvron, H., Martin, L., Stone, K., και άλλοι. “LLaMA: Open and Efficient Foundation Language Models,” *arXiv preprint* arXiv:2302.13971, 2023.

ΠΑΡΑΡΤΗΜΑ Α: Εγχειρίδιο χρήσης της εφαρμογής

Το παρόν παράρτημα περιλαμβάνει βασικές οδηγίες για τη σύνδεση στο σύστημα και τον καθορισμό ρόλων χρηστών, όπως αυτά εφαρμόζονται στην παρούσα εφαρμογή. Οι ρυθμίσεις αυτές επηρεάζουν τη δυνατότητα πρόσβασης και την προβολή εγγράφων και υποθέσεων ανά επίπεδο χρήστη.

🔐 Σύνδεση & ρόλοι

Πρώτη Σύνδεση

Ο διαχειριστής σάς δίνει όνομα χρήστη και προσωρινό κωδικό.

-Συνδεθείτε στο σύστημα και αλλάξτε άμεσα τον κωδικό σας.

-Θα σας ζητηθεί να αλλάξετε κωδικό άμεσα

Ρόλοι Χρηστών

Ο διαχειριστής ορίζει τον ρόλο σας:

-Βασικός χρήστης (Basic): πρόσβαση σε υποθέσεις και έγγραφα.

-Επόπτης (supervisor): βλέπει υποθέσεις του τμήματός του, και τις private των υφισταμένων του.

-Διευθυντής (head): Έχει την ευρύτερη εποπτεία.

-Administrator: πλήρης έλεγχος χρηστών και ρυθμίσεων.

Επιπλέον, κάθε χρήστης ανήκει σε Τμήμα (Department) και Διεύθυνση (Division) που επηρεάζουν την προσβασιμότητα στις υποθέσεις.

📁 Υποθέσεις & Δικαιώματα Πρόσβασης

Δημιουργία Νέας Υπόθεσης.

Από το μενού, επιλέξτε "Νέα Υπόθεση My cases" και συμπληρώστε

-Πρωτόκολλο

-Σημειώσεις

-Επίπεδο ορατότητας

Public Όλοι οι χρήστες

Local Μόνο χρήστες του ίδιου τμήματος

Private Μόνο εσείς και ο Επόπτης σας

Κάθε υπόθεση μπορεί να ενημερωθεί, να αρχειοθετηθεί ή να εμπλουτιστεί με νέα αρχεία.

📄 Έγγραφα σε Υπόθεση

Ανέβασμα Εγγράφων

Μέσα στην υπόθεση: Πατήστε "Ανέβασμα αρχείου".
Επιλέξτε τοπικό αρχείο από τον υπολογιστή σας, ή εισάγετε URL προς έγγραφο στο διαδίκτυο (π.χ. PDF, HTML).

Το σύστημα θα

Αναλύσει αυτόματα το έγγραφο
Θα εξάγει δομημένο κείμενο και μεταδεδομένα

Προβολή & Διαχείριση Εγγράφων σε Υπόθεση

Μπαίνοντας στις λεπτομέρειες μιας υπόθεσης, μπορείτε να δείτε όλα τα σχετικά έγγραφα που έχουν ανέβει καθώς και την κατάσταση επεξεργασίας τους.

Τι εμφανίζεται στην καρτέλα της υπόθεσης

Λίστα Εγγράφων: Όλα τα αρχεία που έχουν ανεβεί (τοπικά ή μέσω URL)
Κατάσταση Επεξεργασίας (Processes): Για κάθε έγγραφο εμφανίζεται το pipeline που εκτελείται (π.χ. parsing, NER, enrichment) και το στάδιο υλοποίησης.
Γνωστό Έγγραφο: Αν το σύστημα αναγνωρίσει το έγγραφο ως "γνωστό", θα σας εμφανίσει και τον τύπο του (π.χ. συμβόλαιο, τιμολόγιο κ.λπ.).

Προβολή Εγγράφου

Κάνοντας κλικ σε ένα έγγραφο, εμφανίζονται τρία στοιχεία: το αρχικό αρχείο (όπως το ανεβάσατε), το εξαγόμενο κείμενο από τη διαδικασία ανάλυσης (π.χ. OCR ή parsing), τα μεταδεδομένα του εγγράφου, π.χ. ημερομηνία δημιουργίας, συγγραφέας, κωδικοποίηση, κ.ά.

Λήψη Δεδομένων

Για κάθε αρχείο μπορείτε: να κατεβάσετε (download) το αρχικό αρχείο, να κατεβάσετε (download) το εξαγόμενο κείμενο σε .txt ή να κατεβάσετε (download) τα μεταδεδομένα σε .json ή .csv .

🧠 Επιβεβαίωση & Επεξεργασία Οντοτήτων (NER Review)

Επιβεβαίωση Οντοτήτων

Για κάθε έγγραφο που ανεβαίνει στην υπόθεση, ο χρήστης οφείλει να επιβεβαιώσει ή να διορθώσει τις οντότητες που εντοπίστηκαν αυτόματα από το σύστημα. Πατήστε REVIEW NER. Αυτό σας μεταφέρει στην ειδική καρτέλα "NER: Annotate & Compare".

Οθόνης NER: Καρτέλες

Editor: Προβολή και επεξεργασία του κειμενου με τις εντοπισμένες οντότητες (ονόματα, IPs, ημερομηνίες κ.λπ.). Μπορείτε να προσθέσετε, αφαιρέσετε ή διορθώσετε οντότητες απευθείας στο κείμενο.
Saved Entities: Πίνακας με τις οντότητες που έχουν αποθηκευτεί. Περιλαμβάνει: ετικέτα (π.χ. IP, ORG), αρχή/τέλος χαρακτήρων, και το απομονωμένο κείμενο.
Model Extracted: Εμφανίζει τις οντότητες που εντόπισε το μοντέλο. Χρήσιμο για να συγκρίνετε με τις αλλαγές σας.
User Approved: Δείχνει τις τελικές οντότητες που έχετε επιβεβαιώσει ως χρήστης.

Πως προσθέτω ή τροποποιώ οντότητα

- Επιλέξτε κείμενο με το ποντίκι
- Επιλέξτε ετικέτα (π.χ. PERSON, IP, DATE)
- Η οντότητα θα επισημανθεί με χρώμα
- ❌ Κουμπί για διαγραφή (ή παρατεταμένο πάτημα για μαζική διαγραφή ίδιων οντοτήτων)

🛠 Εργαλεία (κουμπιά στη γραμμή εργαλείων)

Κουμπί	Περιγραφή
💾 Save Annotations	Αποθηκεύει τις οντότητες που έχετε προσθέσει ή τροποποιήσει
📂 Load Annotations	Φορτώνει προηγούμενες αποθηκευμένες οντότητες (αν υπάρχουν)
🌐 Highlight Foreign Chars	Εντοπίζει και τονίζει ξένους χαρακτήρες (π.χ. αν υπάρχουν μη-ελληνικά σύμβολα σε ελληνικό κείμενο)
🧹 Clean Spacing	Καθαρίζει διπλά ή περιττά κενά μεταξύ των λέξεων.
␣ Show Spaces	Εμφανίζει όλα τα κενά και αλλαγές γραμμής για ακρίβεια
❌ Remove All Entities	Αφαιρεί όλες τις επισημάνσεις/οντότητες από το κείμενο.

📋 Πληροφορίες στο Πάνω Δεξί Μέρος

🧠 Model: Το μοντέλο που χρησιμοποιήθηκε (π.χ. spaCy, gemma, κ.ά.)

🌍 Language: Η γλώσσα του κειμένου όπως ανιχνεύθηκε (π.χ. el, en)

✍️ Επιβεβαίωση και Διόρθωση Οντοτήτων (NER Editor): Όταν ανεβάζετε ένα έγγραφο σε μια υπόθεση, είστε υποχρεωμένοι να επιβεβαιώσετε ή να τροποποιήσετε τις οντότητες που εντοπίστηκαν αυτόματα από το σύστημα.

🌐 Explore Graph – Γραφική Εξερεύνηση Οντοτήτων

Διαδραστικός γράφος που απεικονίζει:

Υποθέσεις
Έγγραφα
Οντότητες (IP, domain, πρόσωπα, ημερομηνίες)

Εργαλεία στη Επάνω Γραμμή Εργαλείων

- 🔍 Αναζήτηση κόμβων με auto-complete
- 📦 Φιλτράρισμα κόμβων κατά τύπο
- ↩️ Undo / ↪️ Redo ενεργειών
- 🔁 Layout επιλογή (κυκλικό, ιεραρχικό, πλέγμα)

📥 Εξαγωγή σε PNG, SVG, JSON

Εργαλεία στη Κάτω Γραμμή Εργαλείων

✍️ **Προσθήκη custom node, edge ή σημείωσης**

Node (Κόμβος): Πατήστε "Add Node" και εισάγετε όνομα, τύπο και ιδιότητες. Ο κόμβος θα προστεθεί στο γράφημα και μπορείτε να τον συνδέσετε με άλλους.

Edge (Ακμή): Επιλέξτε δύο κόμβους (με Ctrl+Click ή Shift+Click) και πατήστε "Add Edge" για να δημιουργήσετε σύνδεση μεταξύ τους. Η ακμή μπορεί να περιλαμβάνει και περιγραφή (label).

Note (Σημείωση): Μπορείτε να προσθέσετε επεξηγηματικό σχόλιο σε οποιοδήποτε σημείο του γραφήματος.

🔥 **Διαγραφή Κόμβων και Ακμών**

Πατήστε Delete για να διαγράψετε τον επιλεγμένο κόμβο ή ακμή
Επιλέξτε πολλούς κόμβους (με Ctrl/Shift) και πατήστε Delete για μαζική διαγραφή.

Εργαλεία Μενού Επιλογών (δεξί κλικ)

📄 Προβολή metadata κόμβου και λήψη JSON
⚙️ Πραγματοποίηση NER σε κόμβο εγγράφου
➕ Expand κόμβου για περισσότερες συνδέσεις

📑 **Γνωστά Έγγραφα (Known Documents)**

Αν το σύστημα αναγνωρίσει ένα έγγραφο ως γνωστό (γνωρίζει τη δομή του και έχει καθοριστεί parser και template) μπορείτε να ακολουθήσετε την παρακάτω διαδικασία:

- Πατήστε Process Document

Το σύστημα:

Επιλέγει parser
Εξάγει κρίσιμα δεδομένα
Εμπλουτίζει IPs με πρόσθετες πληροφορίες
Παρέχει preview, μεταδεδομένα, και δυνατότητα λήψης DOCX

🕓 **Επιλογή Ζώνης Ώρας & Μορφής**

Κατά την προβολή ή λήψη αναφοράς, μπορείτε να ρυθμίσετε:

🌍 Timezone (π.χ. Europe/Athens, UTC)
🕐 Date Format (π.χ. DD/MM/YYYY HH:mm:ss, YYYY-MM-DD)

Παραδείγματα:

DD/MM/YYYY HH:mm:ss → 05/06/2025 13:45:20
YYYY-MM-DD → 2025-06-05
dddd, MMMM Do YYYY → Thursday, June 5th 2025

💬 RAG BOT – Εικονικός Βοηθός

Πρόσβαση

Από το μενού: CHAT

⚠️ Σημείωση: Οι απαντήσεις ενδέχεται να περιέχουν ανακρίβειες. Πατήστε I Understand για να συνεχίσετε.

Ρυθμίσεις

- 🤖 Model: Επιλέγετε από τα διαθέσιμα μοντέλα LLM
- 🌡️ Temp: Ρυθμίζετε πόσο δημιουργικές είναι οι απαντήσεις (0 = αυστηρά, 1 = πιο ανοιχτές)
- 📚 Top K: Ρυθμίζετε πόσα έγγραφα ανακτώνται και συμπεριλαμβάνονται στο ερώτημά σας προς το LLM

Φίλτρα

- 📁 Επιλογή υποθέσεων
- 📄 Επιλογή εγγράφων

Υποβολή Ερώτησης

- Πληκτρολογήστε ερώτηση

-Πατήστε Enter ή Ask

-Προβάλλονται: Ερώτηση, Πηγές εγγράφων, Απάντηση

Συνεδρία

- ⏳ Μετρητής χρόνου / κουμπί παράτασης του χρόνου συνεδρίας.

📊 Πίνακας Ελέγχου Υποθέσεων (Case Dashboard)

Στατιστικά

-Total Active Cases

-Cases Last 7/30/365 Days

-Μέσος Χρόνος Ολοκλήρωσης (σε ημέρες)

Φίλτρα Ημερομηνίας

-Έναρξη / Λήξη περιόδου

Διαγράμματα

- 🧠 Cases by Status
- 🧠 Cases by Visibility

Ετήσια Προβολή

-Weekly Progress (Απόλυτες Τιμές & Ποσοστά)

🔎 Αναζήτηση Εγγράφων

Η εφαρμογή προσφέρει ένα προηγμένο εργαλείο αναζήτησης που σας επιτρέπει να εντοπίζετε έγγραφα με λέξεις-κλειδιά, σημασιολογική ανάλυση ή συνδυαστικά.

🛠️ Επιλογές Αναζήτησης

1. Ερώτημα (Query)

Στο επάνω πεδίο εισάγετε την αναζήτησή σας ή επιλέγετε μια από τις πρόσφατες (ιστορικό αποθηκεύεται τοπικά).

2. Τύπος Αναζήτησης (Search Type)

Τύπος	Περιγραφή
Keyword Search	Κλασική αναζήτηση με βάση το κείμενο
Semantic Search	Νοηματική αντιστοίχιση ερωτήματος με περιεχόμενο
Combined	Συνδυασμός keyword και semantic αναζήτησης

🧭 Πρόσθετες Επιλογές (για Keyword Search)

Πεδίο **Αναζήτησης(Scope)**

Πεδίο	Αναζητά στο...
Full	Όλο το περιεχόμενο
File Name Only	Μόνο στο όνομα αρχείου
Content Only	Μόνο στο περιεχόμενο
Metadata Only	Μόνο σε μεταδεδομένα (π.χ. συγγραφέας, ημερομηνία)

Αν επιλέξετε Metadata Only, μπορείτε να διαλέξετε και συγκεκριμένο πεδίο (π.χ. author, doctype).

Τύπος Αντιστοίχισης (Match Type)

-Flexible Match: Αντιστοιχίσεις χωρίς ακριβή σύμπτωση

-Exact Match: Ακριβής αντιστοίχιση με τη φράση

📊 Επιλογές για Semantic Search

Αν επιλέξετε Semantic Search, μπορείτε να ορίσετε το πλήθος των αποτελεσμάτων (Top K) από 5 έως 20. Περισσότερα αποτελέσματα ενδέχεται να εμφανίζουν επιπλέον παραλλαγές.

📋 Αποτελέσματα Αναζήτησης

-A/A

-Όνομα Αρχείου (πατήστε για προβολή εγγράφου)

-Snippet (απόσπασμα από το έγγραφο με εμφάνιση των όρων που ταιριάζουν)

-Σκορ (σε semantic αναζητήσεις: πόσο σχετικό θεωρείται το αποτέλεσμα)

Οι λέξεις/φράσεις που εντοπίζονται στο αποτέλεσμα επισημαίνονται έντονα.

📍 Πλοήγηση στα Αποτελέσματα

-Κάθε σελίδα εμφανίζει έως 20 αποτελέσματα

-Χρησιμοποιήστε το pagination κάτω από τα αποτελέσματα για να μεταβείτε σε επόμενη σελίδα

✅ **Συμβουλές Χρήσης**

-Μπορείτε να εισάγετε ερώτηση σε φυσική γλώσσα (π.χ. "έγγραφα με τιμολόγια του 2023")

-Οι semantic αναζητήσεις λειτουργούν καλύτερα με φυσικές ερωτήσεις ή ευρύτερες έννοιες

-Το ιστορικό αναζήτησης αποθηκεύεται τοπικά και μπορείτε να το χρησιμοποιείτε ξανά.

ΠΑΡΑΡΤΗΜΑ Β: Πηγαίος κώδικας εφαρμογής

Ο πηγαίος κώδικας που υλοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας είναι διαθέσιμος μέσω της διαδικτυακής πλατφόρμας GitHub. Το αποθετήριο φιλοξενεί όλα τα βασικά υποσυστήματα της εφαρμογής, τα οποία περιγράφονται αναλυτικά στα επιμέρους κεφάλαια της μελέτης.
Σύνδεσμος αποθετηρίου:
https://github.com/ChristosTheodorou/restart
Για την καταγραφή των στατιστικών του πηγαίου κώδικα χρησιμοποιήθηκε το εργαλείο **tokei**, το οποίο μετρά γραμμές κώδικα, σχόλια και κενές γραμμές ανά γλώσσα προγραμματισμού. Προκειμένου να ληφθεί υπόψη μόνο ο παραγωγικός κώδικας και να αποκλειστούν αυτόματα παραγόμενα, προσωρινά ή βοηθητικά αρχεία (π.χ. migrations, __pycache__, node_modules, αρχεία τεκμηρίωσης κ.λπ.), η εκτέλεση του εργαλείου πραγματοποιήθηκε με την ακόλουθη εντολή:

```
tokei . --exclude "package-lock.json,package.json,yarn.lock,Pipfile.lock,poetry.lock,**/migrations,**/__pycache__,**/node_modules,**/.venv,**/venv,**/env,**/myenv,static/dist,dist,build,staticfiles,media,docs,*.md,*.rst,*.txt,commands.md,todo.md,__old__,services/documentation,.github,.vscode,.idea,*.swp,__snapshots__,mkcert,customocr,services/opensearch/config/custom-opensearch-config,services/tika/tika-dual-emitter,services/tika/config,services/n8n,/models" -o json > stats_clean.json
```

Η ανάλυση έδειξε ότι το έργο αποτελείται από **36.408 συνολικές γραμμές**, εκ των οποίων **28.618 γραμμές κώδικα**, **2.940 γραμμές σχολίων** και **4.850 κενές γραμμές**. Η κυρίαρχη γλώσσα είναι η **Python**, με την πλειοψηφία του κώδικα, ενώ σημαντική παρουσία έχουν επίσης η **JavaScript** και η **HTML**, γεγονός που υποδηλώνει έναν συνδυασμό back-end και front-end ανάπτυξης.

Συνοπτικά αποτελέσματα ανά γλώσσα:

- Python → 16.309 γραμμές κώδικα, 1.349 σχόλια, 2.484 κενές (20.142 συνολικές)
- JavaScript → 5.835 γραμμές κώδικα, 381 σχόλια, 1.002 κενές (7.218 συνολικές)
- HTML → 3.671 γραμμές κώδικα, 51 σχόλια, 697 κενές (4.419 συνολικές)
- CSS → 1.307 γραμμές κώδικα, 38 σχόλια, 393 κενές (1.738 συνολικές)
- YAML → 1.164 γραμμές κώδικα, 520 σχόλια, 165 κενές (1.849 συνολικές)
- JSON → 1.352 γραμμές κώδικα, 0 σχόλια, 95 κενές (1.447 συνολικές)
- Shell (Bash) → 123 γραμμές κώδικα, 172 σχόλια, 14 κενές (309 συνολικές)
- Markdown → 57 γραμμές κώδικα (χωρίς σχόλια), 0 κενές (57 συνολικές)
- Άλλες/διάφορες μικρές → <50 γραμμές η καθεμία

Γενικά σύνολα:

- Συνολικές γραμμές: 36.408
- Κώδικας: 28.618
- Σχόλια: 2.940
- Κενές: 4.850