ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΠΡΟΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

A Cloud Application for Machine Learning Models Management & Execution (Μια Εφαρμογή Νέφους για την Διαχείριση & Εκτέλεση Μοντέλων Μηχανικής Μάθησης)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Νικολάου Σπύρου, Νικολάου Ριζόγιαννη

Επιβλέπων: Αναπλ. Καθηγητής Κυριάκος Κρητικός

Μέλη εξεταστικής επιτροπής: Καθηγητής Σπύρος Κοκολάκης, Αναπλ. Καθηγητής Παναγιώτης Συμεωνίδης

Σάμος, Φεβρουάριος 2026

Πρόλογος και ευχαριστίες

Στο σημείο αυτό, θα θέλαμε να εκφράσουμε τις ευχαριστίες μας σε όλους όσοι συνέβαλαν στην ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Καταρχάς, ευχαριστούμε θερμά τον **Αναπληρωτή Καθηγητή κ. Κυριάκο Κρητικό** για την καθοδήγηση, τις ουσιαστικές παρατηρήσεις και τη συνεχή υποστήριξή του καθ' όλη τη διάρκεια εκπόνησης της εργασίας, καθώς και για τη συμβολή του στη συνολική ακαδημαϊκή μας πορεία στο πλαίσιο του Προπτυχιακού Προγράμματος Σπουδών. Επιπλέον, ευχαριστούμε όλους τους διδάσκοντες του Προγράμματος για τις γνώσεις και τα εφόδια που μας παρείχαν κατά τη διάρκεια των σπουδών μας.

Θα θέλαμε επίσης να ευχαριστήσουμε ο ένας τον άλλον για την άριστη συνεργασία, τον συντονισμό και τη συστηματική ανταλλαγή απόψεων, οι οποίες συνέβαλαν καθοριστικά στην επιτυχή ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Τέλος, ευχαριστούμε τις οικογένειες και το φιλικό μας περιβάλλον για τη στήριξη και την κατανόηση που επέδειξαν καθ' όλη τη διάρκεια της ακαδημαϊκής μας προσπάθειας.

© 2026

Νικόλαος Σπύρου, Νικόλαος Ριζόγιαννης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Πίνακας Περιεχομένων

Περιεχόμενα

Περίληψη 14
Abstract 15
1 Εισαγωγή 16
1.1 Το Πρόβλημα 16
1.2 Συνεισφορά 17
1.3 Δομή της Διπλωματικής 18
2 Θεωρητικό Υπόβαθρο 19
2.1 Μηχανική Μάθηση 19
2.1.1 Μοντέλο Μηχανικής Μάθησης *20*
2.1.2 Βασικές Εφαρμογές της Μηχανικής Μάθησης *20*
2.1.3 Αλγοριθμικές Προσεγγίσεις και Κατηγορίες Μηχανικής Μάθησης *21*
2.2 Κύκλος Ζωής Μοντέλων Μηχανικής Μάθησης 22
2.3 Υπολογιστικό Νέφος 24
2.3.1 Βασικά Χαρακτηριστικά και Πλεονεκτήματα του Υπολογιστικού Νέφους *24*
2.3.2 Μοντέλα Παροχής Υπηρεσιών Υπολογιστικού Νέφους *25*
2.3.3 Τεχνολογίες Απομόνωσης και Διαχείρισης Υπολογιστικών Πόρων *26*
3 Σχετικές Εργασίες 28
3.1 Amazon SageMaker 28
3.2 Google Vertex AI 29
3.3 Microsoft Azure Machine Learning 29
3.4 MLflow 30
3.5 Σύγκριση με την Προτεινόμενη Εφαρμογή 30
4 Ανάπτυξη Εφαρμογής 33
4.1 Απαιτήσεις Συστήματος 33
4.1.1 Λειτουργικές Απαιτήσεις *33*
4.1.2 Μη Λειτουργικές Απαιτήσεις *36*
4.1.3 Τεχνολογικοί Περιορισμοί και Εξαρτήσεις *38*
4.2 Σχεδίαση Συστήματος 39
4.2.1 Διάγραμμα Περιβάλλοντος (System Context Diagram) *39*
4.2.2 Αρχιτεκτονική Συστήματος και Διάγραμμα Συστατικών *41*
4.2.3 Διαγράμματα Περιπτώσεων Χρήσης *44*
4.2.4 Διαγράμματα Ακολουθίας *53*
4.2.5 Διάγραμμα Σχήματος Βάσης Δεδομένων *69*
4.3 Υλοποίηση Εφαρμογής 71
4.3.1 Τεχνολογίες και Πλαίσια Ανάπτυξης *71*
4.3.2 Δομή και Οργάνωση Κώδικα *72*

4.3.3 Υλοποίηση Ροών Εκπαίδευσης και Εκτέλεσης Μοντέλων *78*
4.3.4 Διαχείριση Αποθήκευσης και Δεδομένων *80*
4.3.5 Χειρισμός Σφαλμάτων και Ζητήματα Ασφάλειας *81*
4.4 Δοκιμές 82
4.5 Ικανοποίηση Απαιτήσεων 84
4.5.1 Μεθοδολογία Αξιολόγησης Απαιτήσεων *84*
4.5.2 Ανάλυση Πλήρους Ικανοποίησης Απαιτήσεων *85*
4.5.3 Ικανοποίηση Λειτουργικών Απαιτήσεων *85*
4.5.4 Ικανοποίηση Μη Λειτουργικών Απαιτήσεων *88*
4.5.5 Περιπτώσεις Μερικής Ικανοποίησης και Αιτιολόγηση *90*
4.5.6 Συνολική Αποτίμηση *93*
5 Εγκατάσταση & Επίδειξη της Εφαρμογής **94**
5.1 Οδηγίες Εγκατάστασης και Εκτέλεσης της Εφαρμογής 94
5.1.1 Απαιτούμενο Λογισμικό και Περιβάλλον Εκτέλεσης *94*
5.1.2 Διαδικασία Εγκατάστασης *95*
5.1.3 Εκκίνηση και Τερματισμός της Εφαρμογής *96*
5.2 Επίδειξη της Εφαρμογής 97
5.2.1 Σενάριο 1: Εκπαίδευση Μοντέλου με Προκαθορισμένο Αλγόριθμο (Weka) *97*
5.2.2 Σενάριο 2: Δημιουργία Προσαρμοσμένου Αλγορίθμου *102*
5.2.3 Σενάριο 3: Εκπαίδευση με Προσαρμοσμένο Αλγόριθμο (Docker-based) *105*
5.2.4 Σενάριο 4: Εκτέλεση Πρόβλεψης με Εκπαιδευμένο Μοντέλο *110*
5.2.5 Σενάριο 5: Διαχείριση Λογαριασμού Χρήστη *114*
5.2.6 Σενάριο 6: Διαχείριση Κατηγοριών *122*
5.2.7 Σενάριο 7: Επανεκπαίδευση Μοντέλων *128*
5.2.8 Σενάριο 8: Διακοπή Εκπαίδευσης/Πρόβλεψης *130*
5.2.9 Σενάριο 9: Μεταφόρτωση και Διαχείριση Δεδομένων *133*
5.3 Αποτίμηση της Εφαρμογής 136
6 Συμπεράσματα και Μελλοντική Εργασία **137**
6.1 Συμπεράσματα 137
6.2 Εμπειρίες από την Υλοποίηση 138
6.3 Περιορισμοί και Σημεία Βελτίωσης 138
6.4 Μελλοντική Εργασία και Όραμα Εξέλιξης 139
Βιβλιογραφία **142**

Λίστα Σχημάτων

Εικόνα 4.1: Διάγραμμα Περιβάλλοντος 41

Εικόνα 4.2: Διάγραμμα Συστατικών 45

Εικόνα 4.3: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Συνόλου Δεδομένων 47

Εικόνα 4.4: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Αλγορίθμων 49

Εικόνα 4.5: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Μοντέλων 51

Εικόνα 4.6: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Κατηγοριών 52

Εικόνα 4.7: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Χρηστών 54

Εικόνα 4.8: Διάγραμμα Ακολουθίας - Εγγραφή Χρήστη 56

Εικόνα 4.9: Διάγραμμα Ακολουθίας - Αυθεντικοποίηση Χρήστη 58

Εικόνα 4.10: Διάγραμμα Ακολουθίας - Μεταφόρτωση Συνόλου Δεδομένων 59

Εικόνα 4.11: Διάγραμμα Ακολουθίας - Διαμοιρασμός Πόρων 60

Εικόνα 4.12: Διάγραμμα Ακολουθίας - Διαχείριση Κατηγοριών 62

Εικόνα 4.13: Διάγραμμα Ακολουθίας - Εκπαίδευση Προκαθορισμένου Αλγόριθμου 64

Εικόνα 4.14: Διάγραμμα Ακολουθίας - Εκπαίδευση Προσαρμοσμένου Αλγόριθμου Εκτέλεση Πρόβλεψης 66

Εικόνα 4.15: Διάγραμμα Ακολουθίας - Εκτέλεση Μοντέλου 68

Εικόνα 4.16: Διάγραμμα Σχήματος Βάσης Δεδομένων 70

Εικόνα 4.17: Διάγραμμα Πακέτων Backend 74

Εικόνα 4.18: Διάγραμμα Πακέτων Frontend 75

Εικόνα 4.19: Διάγραμμα Κλάσεων 78

Εικόνα 5.1: Μενού Πλοήγησης - Επιλογή Εκπαιδεύσεων 99

Εικόνα 5.2: Επιλογή Weka Εκπαίδευσης 99

Εικόνα 5.3: Ενσωμάτωση Αλγορίθμου και Παραμέτρων 100

Εικόνα 5.4: Παραμετροποίηση Συνόλου Δεδομένων & Εκκίνηση Εκπαίδευσης 100

Εικόνα 5.5: Weka Training Kubernetes Pod (K9S) 101

Εικόνα 5.6: Διαγράμματα Μετρικών 101

Εικόνα 5.7: Πάνελ Ενεργών Εκπαιδεύσεων 102

Εικόνα 5.8: Μενού Πλοήγησης - Επιλογή Αλγορίθμων 103

Εικόνα 5.9: Σελίδα Αλγορίθμων 103

Εικόνα 5.10: Ανέβασμα Προσαρμοσμένου Αλγορίθμου 104

Εικόνα 5.11: Πάνελ Προσαρμοσμένων Αλγορίθμων 104
Εικόνα 5.12: Μενού Πλοήγησης - Επιλογή Εκπαιδεύσεων 106
Εικόνα 5.13: Σελίδα Επιλογής Είδους Εκπαίδευσης 106
Εικόνα 5.14: Επιλογή Προσαρμοσμένου Αλγορίθμου 107
Εικόνα 5.15: Επιλογή Συνόλου Δεδομένων από τα Υπάρχοντα 107
Εικόνα 5.16: Εκκίνηση Εκπαίδευσης Προσαρμοσμένου Αλγορίθμου 108
Εικόνα 5.17: Πάνελ Ενεργών Επανεκπαιδεύσεων 108
Εικόνα 5.18: Λίστα Εκπαιδεύσεων 109
Εικόνα 5.19: Λίστα Εκπαιδευμένων Μοντέλων 109
Εικόνα 5.20: Λήψη Εκπαιδευμένου Μοντέλου 110
Εικόνα 5.21: Μενού Πλοήγησης - Επιλογή Εκτελέσεων 111
Εικόνα 5.22: Σελίδα Επιλογών Εκτέλεσης 111
Εικόνα 5.23: Εκτέλεση Προσαρμοσμένου Μοντέλου 112
Εικόνα 5.24: Επιλογή Εκπαιδευμένου Μοντέλου 112
Εικόνα 5.25: Επιλογή Συνόλου Δεδομένων Πρόβλεψης από τα Υπάρχοντα 113
Εικόνα 5.26: Ενεργές Εκτελέσεις Μοντέλων 113
Εικόνα 5.27: Λήψη Αποτελεσμάτων Εκτέλεσης 114
Εικόνα 5.28: Πλοήγηση στην Εγγραφή Χρήστη 116
Εικόνα 5.29: Εγγραφή Χρήστη 116
Εικόνα 5.30: Πλοήγηση στη Σύνδεση Χρήστη 117
Εικόνα 5.31: Αυθεντικοποίηση Χρήστη 117
Εικόνα 5.32: Σελίδα Καλωσορίσματος Χρήστη 118
Εικόνα 5.33: Πλοήγηση στην Αλλαγή Κωδικού Πρόσβασης 119
Εικόνα 5.34: Αλλαγή Κωδικού Πρόσβασης 119
Εικόνα 5.35: Πλοήγηση στην Ανάκτηση Κωδικού Πρόσβασης 120
Εικόνα 5.36: Αίτηση Επαναφοράς Κωδικού Πρόσβασης 121
Εικόνα 5.37: Περιεχόμενο Μηνύματος Επαναφοράς Κωδικού 121
Εικόνα 5.38: Επαναφορά Κωδικού Πρόσβασης 122
Εικόνα 5.39: Μενού Πλοήγησης - Επιλογή Κατηγοριών 124
Εικόνα 5.40: Σελίδα Κατηγοριών 124
Εικόνα 5.41: Πλοήγηση στη Δημιουργία Κατηγορίας 124

Εικόνα 5.42: Αίτημα Δημιουργίας Κατηγορίας (User) 125
Εικόνα 5.43: Σελίδα Διαχείρισης Κατηγοριών (Pending) 125
Εικόνα 5.44: Μενού Πλοήγησης - Επιλογή Διαχείρισης Κατηγοριών 126
Εικόνα 5.45: Αποδοχή/Απόρριψη Αιτήματος Δημιουργίας Κατηγορίας (Admin) 126
Εικόνα 5.46: Σελίδα Διαχείρισης Κατηγοριών (Approved) 127
Εικόνα 5.47: Σελίδα Διαχείρισης Κατηγοριών (Rejected) 127
Εικόνα 5.48: Επιλογή Τύπου Εκπαίδευσης 129
Εικόνα 5.49: Επανεκπαίδευση Προσαρμοσμένου Αλγορίθμου 129
Εικόνα 5.50: Kubernetes Pod Επανεκπαίδευση 130
Εικόνα 5.51: Διακοπή Εκπαίδευσης 131
Εικόνα 5.52: Διακοπή Πρόβλεψης 131
Εικόνα 5.53: Μήνυμα Επιτυχίας Διακοπής Πρόβλεψης 132
Εικόνα 5.54: Διακοπή Εκπαίδευσης K8s Logs 132
Εικόνα 5.55: Μενού Πλοήγησης - Διαχείριση Συνόλου Δεδομένων 133
Εικόνα 5.56: Μεταφόρτωση Συνόλου Δεδομένων 134
Εικόνα 5.57: Μεταφορτωμένα Σύνολα Δεδομένων 134

Λίστα Πινάκων

Πίνακας 3.1 Συνοπτική Σύγκριση Παρόμοιων Συστημάτων 32

Ακρωνύμια

Ακρωνύμιο	Περιγραφή
AI	Artificial Intelligence
API	Application Programming Interface
ARFF	Attribute-Relation File Format
AWS	Amazon Web Services
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSV	Comma-Separated Values
DB	Database
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
JPA	Java Persistence API
JWT	JSON Web Token
K8s	Kubernetes
LTS	Long-Term Support
ML	Machine Learning
PaaS	Platform as a Service
PVC	Persistent Volume Claim

RAM	Random Access Memory
RBAC	Role-Based Access Control
REST	Representational State Transfer
S3	Simple Storage Service
SaaS	Software as a Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
UI	User Interface

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τον σχεδιασμό και την υλοποίηση μιας εφαρμογής υπολογιστικού νέφους για τη δημιουργία, εκπαίδευση, διαχείριση και εκτέλεση μοντέλων μηχανικής μάθησης σε πολυχρηστικό περιβάλλον. Η εφαρμογή υποστηρίζει τόσο προκαθορισμένους αλγορίθμους της βιβλιοθήκης Weka όσο και προσαρμοσμένους αλγορίθμους που παρέχονται από τους χρήστες ως δοχειοποιημένες εφαρμογές (Docker images), επιτρέποντας την ενσωμάτωση αλγορίθμων σε οποιαδήποτε γλώσσα προγραμματισμού.

Η αρχιτεκτονική βασίζεται σε πολυεπίπεδο διαχωρισμό ευθυνών, με backend υλοποιημένο σε Java/Spring Boot, frontend ως single-page application σε TypeScript, σχεσιακή βάση δεδομένων PostgreSQL για μεταδεδομένα και αποθετήριο αντικειμένων MinIO για αρχεία. Οι εργασίες εκπαίδευσης και πρόβλεψης εκτελούνται ασύγχρονα ως απομονωμένα Kubernetes Jobs, εξασφαλίζοντας απομόνωση πόρων, ασφάλεια εκτέλεσης και ανεξαρτησία από τον πάροχο νέφους.

Η εφαρμογή παρέχει πλήρη διαχείριση του κύκλου ζωής μοντέλων, ιεραρχική κατηγοριοποίηση πόρων με ροή έγκρισης, μηχανισμούς ασφαλείας (JWT με RSA, RBAC, Argon2id), δυνατότητα επανεκπαίδευσης, διαμοιρασμό πόρων μεταξύ χρηστών και παρακολούθηση κατάστασης εργασιών σε πραγματικό χρόνο. Η αξιολόγηση του συστήματος μέσω αυτοματοποιημένων και χειρωνακτικών δοκιμών επιβεβαιώνει την πλήρη ικανοποίηση των κρίσιμων λειτουργικών απαιτήσεων, ενώ αναγνωρίζονται σημεία μερικής ικανοποίησης ορισμένων απαιτήσεων που αποτελούν αντικείμενο μελλοντικής βελτίωσης.

Συνολικά, η εργασία καταδεικνύει ότι είναι εφικτή η ανάπτυξη μιας επεκτάσιμης, ασφαλούς και τεχνολογικά ανεξάρτητης πλατφόρμας μηχανικής μάθησης, κατάλληλης τόσο για εκπαιδευτικά και ερευνητικά περιβάλλοντα όσο και ως βάση για μελλοντική εξέλιξη προς παραγωγικές κατευθύνσεις.

Λέξεις-κλειδιά: Μηχανική Μάθηση, Υπολογιστικό Νέφος, Δοχειοποίηση, Kubernetes, Ασύγχρονη Εκτέλεση, Docker, Εκπαίδευση Μοντέλων, RESTful API, Spring Boot, Πολυχρηστικό Περιβάλλον

Abstract

This thesis presents the design and implementation of a cloud application for the creation, training, management, and execution of machine learning models in a multi-user environment. The application supports both predefined algorithms from the Weka library and custom algorithms provided by users as containerized applications (Docker images), enabling the integration of algorithms implemented in any programming language.

The architecture is based on a multi-layered separation of concerns, with a backend implemented in Java/Spring Boot, a frontend as a single-page application in TypeScript, a PostgreSQL relational database for metadata, and a MinIO object storage for files. Training and prediction tasks are executed asynchronously as isolated Kubernetes Jobs, ensuring resource isolation, execution security, and cloud provider independence.

The application provides full model lifecycle management, hierarchical resource categorization with an approval workflow, security mechanisms (JWT with RSA, RBAC, Argon2id), retraining capabilities, resource sharing between users, and real-time task status monitoring. System evaluation through automated and manual testing confirms the full satisfaction of critical functional requirements, while areas of partial satisfaction of some requirements are identified as subjects for future improvement.

Overall, this work demonstrates that it is feasible to develop a scalable, secure, and technology-independent machine learning platform, suitable for both educational and research environments as well as a foundation for future evolution towards production-grade directions.

Keywords: Machine Learning, Cloud Computing, Containerization, Kubernetes, Asynchronous Execution, Docker, Model Training, RESTful API, Spring Boot, Multi-user Environment

1 Εισαγωγή

1.1 Το Πρόβλημα

Η ραγδαία εξέλιξη της μηχανικής μάθησης (Machine Learning - ML) έχει οδηγήσει στη δημιουργία πληθώρας αλγορίθμων και τεχνικών, των οποίων η αξιοποίηση προϋποθέτει σημαντικούς υπολογιστικούς πόρους και σύνθετη υποδομή. Η εκπαίδευση και εκτέλεση τέτοιων μοντέλων αποτελεί συχνά χρονοβόρα διαδικασία, ακατάλληλη για σύγχρονη επεξεργασία αιτημάτων, ενώ η υποστήριξη προσαρμοσμένων αλγορίθμων εισάγει επιπλέον προκλήσεις που σχετίζονται με την ασφάλεια, την απομόνωση και τη διαχείριση πόρων. Επιπλέον, ο χώρος της μηχανικής μάθησης χαρακτηρίζεται από αυξημένες απαιτήσεις σε υπολογιστικούς πόρους, πολυπλοκότητα υλοποίησης και ανάγκη υποστήριξης πολλαπλών χρηστών και ετερογενών αλγορίθμων. Παράλληλα, η αυξανόμενη χρήση δεδομένων μεγάλης κλίμακας (Big Data) καθιστά αναγκαία τη χρήση υποδομών νέφους που προσφέρουν κλιμακωσιμότητα (scalability), απομόνωση και αποδοτική διαχείριση πόρων.

Ωστόσο, τα υπάρχοντα συστήματα και πλατφόρμες μηχανικής μάθησης παρουσιάζουν σειρά ζητημάτων που περιορίζουν τη χρηστικότητα και την ευελιξία τους. Συγκεκριμένα, οι κυρίαρχες εμπορικές πλατφόρμες (Amazon SageMaker, Google Vertex AI, Microsoft Azure ML) είναι στενά συνδεδεμένες με τα αντίστοιχα οικοσυστήματα παρόχων νέφους, δημιουργώντας συνθήκες εγκλωβισμού (vendor lock-in) που δυσχεραίνουν τη μεταφερσιμότητα εφαρμογών και δεδομένων σε εναλλακτικές υποδομές. Η εξάρτηση αυτή καθιστά δύσκολη τη χρήση τους σε ιδιωτικά ή υβριδικά περιβάλλοντα νέφους, ενώ συνεπάγεται σημαντικό λειτουργικό κόστος λόγω των σύνθετων μοντέλων χρέωσης.

Παράλληλα, η υποστήριξη προσαρμοσμένων αλγορίθμων στις εμπορικές πλατφόρμες είναι περιορισμένη ή επιβάλλει συμμόρφωση με αυστηρές τεχνολογικές συμβάσεις και συγκεκριμένες γλώσσες προγραμματισμού ή βιβλιοθήκες, αφαιρώντας από τον χρήστη την ελευθερία πειραματισμού με ετερογενείς υλοποιήσεις. Η απαίτηση αυτή αποτελεί ιδιαίτερα σημαντικό εμπόδιο σε ερευνητικά και εκπαιδευτικά περιβάλλοντα, όπου η δοκιμή νέων αλγοριθμικών προσεγγίσεων αποτελεί κεντρική ανάγκη.

Από την άλλη πλευρά, εργαλεία ανοικτού κώδικα, όπως το MLflow, εστιάζουν κυρίως στην καταγραφή πειραμάτων και την εκδοσιοποίηση μοντέλων, χωρίς να παρέχουν ενσωματωμένους μηχανισμούς εκτέλεσης και ενορχήστρωσης υπολογιστικών φορτίων, διαχείρισης πολυχρηστικών σεναρίων ή απομόνωσης εκτέλεσης κώδικα. Κατά συνέπεια, ο χρήστης υποχρεώνεται να αναλάβει ο ίδιος τη διαχείριση της υποδομής εκτέλεσης, γεγονός που αυξάνει σημαντικά το τεχνικό φράγμα εισόδου.

Επιπρόσθετα, τα υφιστάμενα συστήματα συχνά δεν καλύπτουν ολοκληρωμένα τον κύκλο ζωής των μοντέλων μηχανικής μάθησης — από τη μεταφόρτωση δεδομένων και την εκπαίδευση, έως την αξιολόγηση, τον διαμοιρασμό και την εκτέλεση προβλέψεων — σε μία ενιαία πλατφόρμα. Η κατακερματισμένη αυτή προσέγγιση δυσχεραίνει την αναπαραγωγιμότητα πειραμάτων, τη συνεργασία μεταξύ χρηστών και τη διαφάνεια της διαδικασίας εκπαίδευσης, στοιχεία κρίσιμα τόσο για ακαδημαϊκά όσο και για παραγωγικά περιβάλλοντα.

Τέλος, η πλειονότητα των υπαρχόντων λύσεων απευθύνεται σε εταιρικά περιβάλλοντα μεγάλης κλίμακας, χωρίς να λαμβάνει υπόψη τις ιδιαίτερες ανάγκες εκπαιδευτικών και ερευνητικών πλαισίων, όπου η κατανόηση της εσωτερικής λειτουργίας, η ευκολία χρήσης χωρίς εξειδικευμένη γνώση υποδομών και η δυνατότητα πειραματισμού αποτελούν πρωταρχικές απαιτήσεις.

1.2 Συνεισφορά

Η εργασία αυτή προσεγγίζει το πρόβλημα προτείνοντας μια ενιαία πλατφόρμα, η οποία συνδυάζει τεχνολογίες υπολογιστικού νέφους, δοχειοποίησης και ασύγχρονης εκτέλεσης, παρέχοντας ένα ασφαλές και επεκτάσιμο περιβάλλον για την πειραματική και παραγωγική χρήση αλγορίθμων μηχανικής μάθησης.

Η κύρια συνεισφορά της παρούσας διπλωματικής εργασίας έγκειται στον σχεδιασμό και την υλοποίηση μιας ολοκληρωμένης εφαρμογής υπολογιστικού νέφους (cloud application), η οποία επιτρέπει τη δημιουργία και εκτέλεση μοντέλων μηχανικής μάθησης με τρόπο ασφαλή, ασύγχρονο και επεκτάσιμο. Η εφαρμογή αυτή υποστηρίζει τόσο προκαθορισμένους αλγορίθμους μηχανικής μάθησης, ενσωματωμένους στο backend της, όσο και προσαρμοσμένους αλγορίθμους που παρέχονται από τους χρήστες ως δοχειοποιημένες εφαρμογές (containerised applications).

Ιδιαίτερη έμφαση δίνεται στην απομόνωση της εκτέλεσης των προσαρμοσμένων αλγορίθμων μέσω τεχνολογιών δοχειοποίησης (containerisation) και ενορχήστρωσης (orchestration), καθώς και στην ασύγχρονη διαχείριση των διαδικασιών εκπαίδευσης των αλγορίθμων και πρόβλεψης με βάση τα μοντέλα που παράγονται από αυτούς. Παράλληλα, η εφαρμογή παρέχει πλήρη παρακολούθηση κατάστασης, καταγραφή αποτελεσμάτων και διαχείριση του κύκλου ζωής των μοντέλων, καλύπτοντας βασικές ανάγκες ενός πολυχρηστικού περιβάλλοντος μηχανικής μάθησης.

Από πλευράς πλεονεκτημάτων, η εφαρμογή προσφέρει σε χρήστες και οργανισμούς ένα ενιαίο περιβάλλον που καλύπτει ολόκληρο τον κύκλο ζωής της μηχανικής μάθησης — από τη μεταφόρτωση δεδομένων έως την εκπαίδευση, αξιολόγηση και εκτέλεση μοντέλων — χωρίς να απαιτεί εξειδικευμένη γνώση υποδομών νέφους ή ενορχήστρωσης. Σε αντίθεση με υπάρχουσες πλατφόρμες που δεσμεύουν τον χρήστη σε συγκεκριμένες βιβλιοθήκες ή γλώσσες προγραμματισμού, η προτεινόμενη εφαρμογή επιτρέπει την ενσωμάτωση αλγορίθμων σε οποιαδήποτε τεχνολογία μέσω δοχειοποίησης, εξασφαλίζοντας πλήρη τεχνολογική ανεξαρτησία. Παράλληλα, η ασύγχρονη εκτέλεση σε απομονωμένα δοχεία (containers) εγγυάται ότι χρονοβόρες εκπαιδεύσεις δεν επηρεάζουν την απόκριση του συστήματος, ενώ η δυνατότητα παρακολούθησης και ακύρωσης εργασιών σε πραγματικό χρόνο παρέχει πλήρη έλεγχο στον χρήστη. Τέλος, η χρήση αποκλειστικά εργαλείων ανοικτού κώδικα (Docker, Kubernetes, MinIO) αποτρέπει τον εγκλωβισμό σε συγκεκριμένο πάροχο (vendor lock-in) και επιτρέπει τη μεταφερσιμότητα της πλατφόρμας σε οποιοδήποτε περιβάλλον υποστηρίζει Kubernetes.

1.3 Δομή της Διπλωματικής

Η υπόλοιπη δομή της αναφοράς οργανώνεται ως εξής. Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο της εργασίας που σχετίζεται με τη μηχανική μάθηση, το υπολογιστικό νέφος και τις τεχνολογίες δοχειοποίησης και ενορχήστρωσης. Στο Κεφάλαιο 3 αναλύονται σχετικές εργασίες και συστήματα και πραγματοποιείται σύγκρισή τους με το προτεινόμενο σύστημα. Στο Κεφάλαιο 4 περιγράφεται αναλυτικά η διαδικασία ανάπτυξης της εφαρμογής. Στο Κεφάλαιο 5 παρουσιάζεται η επίδειξη και η αξιολόγηση της εφαρμογής. Τέλος, στο Κεφάλαιο 6 συνοψίζονται τα συμπεράσματα και προτείνονται κατευθύνσεις για μελλοντική εργασία.

2 Θεωρητικό Υπόβαθρο

Το παρόν κεφάλαιο παρουσιάζει το θεωρητικό υπόβαθρο που υποστηρίζει τον σχεδιασμό και την υλοποίηση της εφαρμογής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας. Σκοπός του είναι η εισαγωγή των βασικών εννοιών και τεχνολογιών που σχετίζονται με τη μηχανική μάθηση, τον κύκλο ζωής των μοντέλων, καθώς και το υπολογιστικό νέφος και τις τεχνολογίες δοχειοποίησης και ενορχήστρωσης.

Αρχικά, παρουσιάζονται οι θεμελιώδεις αρχές της μηχανικής μάθησης και οι βασικές κατηγορίες αλγορίθμων, ώστε να τεθεί το απαραίτητο εννοιολογικό πλαίσιο. Στη συνέχεια, αναλύεται ο κύκλος ζωής των μοντέλων μηχανικής μάθησης, ο οποίος αποτελεί κεντρικό άξονα στον σχεδιασμό σύγχρονων πλατφορμών ML. Ακολουθεί εισαγωγή στις έννοιες του υπολογιστικού νέφους, στα μοντέλα παροχής υπηρεσιών και στις τεχνολογίες εικονικοποίησης, δοχειοποίησης και ενορχήστρωσης, οι οποίες αποτελούν κρίσιμα δομικά στοιχεία της αρχιτεκτονικής της προτεινόμενης εφαρμογής.

2.1 Μηχανική Μάθηση

Η μηχανική μάθηση αποτελεί έναν από τους βασικότερους υποτομείς της τεχνητής νοημοσύνης και εστιάζει στη σχεδίαση αλγορίθμων που επιτρέπουν στα υπολογιστικά συστήματα να εξάγουν γνώση από δεδομένα και να βελτιώνουν τη συμπεριφορά τους βάσει εμπειρίας. Σε αντίθεση με τις παραδοσιακές προσεγγίσεις προγραμματισμού, όπου οι κανόνες ορίζονται ρητά από τον προγραμματιστή, στη μηχανική μάθηση οι κανόνες αυτοί προκύπτουν έμμεσα μέσω της ανάλυσης παραδειγμάτων.

Η ευρεία υιοθέτηση της μηχανικής μάθησης σε επιστημονικές και επιχειρησιακές εφαρμογές καθιστά απαραίτητη την κατανόηση των βασικών εννοιών που τη διέπουν, όπως η έννοια του μοντέλου μηχανικής μάθησης, οι κύριες εφαρμογές της, καθώς και οι βασικές κατηγορίες και αλγοριθμικές προσεγγίσεις που χρησιμοποιούνται.

Οι βασικές κατηγορίες μηχανικής μάθησης περιλαμβάνουν την επιβλεπόμενη, την μη επιβλεπόμενη και την ενισχυτική μάθηση. Στην επιβλεπόμενη μάθηση, τα μοντέλα

εκπαιδεύονται με δεδομένα που περιλαμβάνουν γνωστές ετικέτες εξόδου και χρησιμοποιούνται κυρίως για προβλήματα ταξινόμησης (classification) και παλινδρόμησης (regression). Στη μη επιβλεπόμενη μάθηση, τα δεδομένα δεν συνοδεύονται από ετικέτες και ο στόχος είναι η ανακάλυψη της εσωτερικής δομής των δεδομένων, όπως στην ομαδοποίηση (clustering).

Στο πλαίσιο της παρούσας διπλωματικής, οι παραπάνω κατηγορίες αξιοποιούνται άμεσα, καθώς η εφαρμογή υποστηρίζει τόσο αλγορίθμους ταξινόμησης και παλινδρόμησης όσο και αλγορίθμους ομαδοποίησης. Η ανάγκη υποστήριξης διαφορετικών τύπων αλγορίθμων επηρέασε τον σχεδιασμό της εφαρμογής, οδηγώντας στην υλοποίηση μηχανισμών προεπεξεργασίας δεδομένων, παραμετροποίησης και αποθήκευσης αποτελεσμάτων, όπως αναλύεται στο Κεφάλαιο 4.

2.1.1 Μοντέλο Μηχανικής Μάθησης

Ένα μοντέλο μηχανικής μάθησης αποτελεί την υπολογιστική αναπαράσταση της γνώσης που αποκτάται κατά τη διαδικασία εκπαίδευσης ενός αλγορίθμου πάνω σε δεδομένα. Τυπικά, το μοντέλο μπορεί να θεωρηθεί ως μια μαθηματική συνάρτηση ή δομή, η οποία χαρτογραφεί σύνολα δεδομένων εισόδου σε αντίστοιχες εξόδους, βασιζόμενη σε πρότυπα και συσχετίσεις που έχουν εξαχθεί από το εκπαιδευτικό σύνολο.

Κατά την εκπαίδευση, ο αλγόριθμος προσαρμόζει τις εσωτερικές παραμέτρους του μοντέλου με στόχο τη βελτιστοποίηση ενός κριτηρίου απόδοσης, όπως η ελαχιστοποίηση του σφάλματος πρόβλεψης. Το παραγόμενο μοντέλο μπορεί στη συνέχεια να χρησιμοποιηθεί για την παραγωγή προβλέψεων ή αποφάσεων σε νέα, άγνωστα δεδομένα.

Το μοντέλο αποτελεί τον κεντρικό άξονα ολόκληρου του κύκλου ζωής της μηχανικής μάθησης, καθώς είναι το αντικείμενο που αποθηκεύεται, αξιολογείται, διατίθεται για χρήση και επανεκπαιδεύεται με την πάροδο του χρόνου.

2.1.2 Βασικές Εφαρμογές της Μηχανικής Μάθησης

Η μηχανική μάθηση έχει βρει εκτεταμένη εφαρμογή σε πλήθος επιστημονικών, επιχειρησιακών και τεχνολογικών πεδίων, προσφέροντας αυτοματοποιημένες λύσεις σε προβλήματα που χαρακτηρίζονται από πολυπλοκότητα, μεγάλο όγκο δεδομένων και ανάγκη προσαρμογής σε μεταβαλλόμενες συνθήκες (Jordan, 2015). Οι εφαρμογές της

μηχανικής μάθησης διαφοροποιούνται ανάλογα με το είδος του προβλήματος, τη φύση των δεδομένων και τις απαιτήσεις ακρίβειας, ερμηνευσιμότητας και απόδοσης.

Μία από τις σημαντικότερες εφαρμογές της μηχανικής μάθησης αφορά τα προβλήματα πρόβλεψης και ταξινόμησης, όπου το σύστημα καλείται να εκτιμήσει την τιμή ή την κατηγορία ενός άγνωστου χαρακτηριστικού βάσει ιστορικών δεδομένων. Τέτοιες εφαρμογές συναντώνται σε τομείς όπως η πρόβλεψη οικονομικών δεικτών, η ανίχνευση απάτης, η ιατρική διάγνωση και η ανάλυση κινδύνου (Hastie, 2009). Η αξιοπιστία των προβλέψεων αυτών εξαρτάται σε μεγάλο βαθμό από την ποιότητα των δεδομένων και τη σωστή επιλογή του εκπαιδευμένου μοντέλου.

Ένα δεύτερο βασικό πεδίο εφαρμογής είναι η ανάλυση και εξερεύνηση δεδομένων, όπου η μηχανική μάθηση χρησιμοποιείται για την ανακάλυψη κρυφών δομών, προτύπων και συσχετίσεων μέσα σε σύνολα δεδομένων (Jordan, 2015). Οι τεχνικές αυτές αξιοποιούνται ευρέως σε σενάρια επιχειρηματικής ευφυΐας, επιστημονικής έρευνας και ανάλυσης μεγάλου όγκου δεδομένων, επιτρέποντας την εξαγωγή χρήσιμων συμπερασμάτων χωρίς την ύπαρξη προκαθορισμένων ετικετών.

Ιδιαίτερη σημασία έχουν επίσης τα συστήματα συστάσεων, τα οποία βασίζονται στη μηχανική μάθηση για την παροχή προσωποποιημένων προτάσεων προς τους χρήστες (Ricci, 2015). Τέτοιες εφαρμογές χρησιμοποιούνται ευρέως σε πλατφόρμες ηλεκτρονικού εμπορίου, υπηρεσίες ψυχαγωγίας και κοινωνικά δίκτυα, με στόχο τη βελτίωση της εμπειρίας χρήστη και την αύξηση της αποτελεσματικότητας των παρεχόμενων υπηρεσιών.

Τέλος, η μηχανική μάθηση αξιοποιείται σε εφαρμογές λήψης αποφάσεων και αυτοματοποίησης διαδικασιών, όπου τα συστήματα καλούνται να επιλέξουν ενέργειες ή στρατηγικές μέσω συνεχούς αλληλεπίδρασης με το περιβάλλον (Sutton, 2018). Τέτοιες προσεγγίσεις συναντώνται στη ρομποτική, στη βελτιστοποίηση πόρων, στα αυτόνομα συστήματα και στη διαχείριση πολύπλοκων διεργασιών. Οι εφαρμογές αυτές αναδεικνύουν την ανάγκη ύπαρξης ευέλικτων και επεκτάσιμων πλατφορμών που να υποστηρίζουν διαφορετικά είδη μοντέλων και ροών εκτέλεσης, όπως η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας.

2.1.3 Αλγοριθμικές Προσεγγίσεις και Κατηγορίες Μηχανικής Μάθησης

Οι τεχνικές μηχανικής μάθησης διακρίνονται σε τρεις βασικές κατηγορίες: την επιβλεπόμενη μάθηση (supervised learning), τη μη επιβλεπόμενη μάθηση (unsupervised learning) και την ενισχυτική μάθηση (reinforcement learning). Κάθε κατηγορία απευθύνεται

σε διαφορετικά είδη προβλημάτων και απαιτεί διαφορετικές προσεγγίσεις ως προς τη μορφή των δεδομένων και τον τρόπο εκπαίδευσης των μοντέλων.

Στην **επιβλεπόμενη μάθηση**, τα δεδομένα εκπαίδευσης συνοδεύονται από γνωστές ετικέτες εξόδου και ο στόχος είναι η εκμάθηση μιας συνάρτησης που αντιστοιχίζει τις εισόδους στις αντίστοιχες εξόδους. Οι πιο διαδεδομένες εφαρμογές της επιβλεπόμενης μάθησης είναι η ταξινόμηση και η παλινδρόμηση. Ενδεικτικοί αλγόριθμοι αυτής της κατηγορίας είναι οι k-Πλησιέστεροι Γείτονες (k-Nearest Neighbors), τα Δέντρα Απόφασης (Decision Trees), οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) και η Γραμμική Παλινδρόμηση (Linear Regression). Οι αλγόριθμοι αυτοί διαφοροποιούνται ως προς την ακρίβεια, την ικανότητα γενίκευσης, την ανθεκτικότητα στον θόρυβο και την ερμηνευσιμότητα των αποτελεσμάτων τους.

Η **μη επιβλεπόμενη μάθηση** εφαρμόζεται σε δεδομένα χωρίς γνωστές ετικέτες εξόδου και στοχεύει στην ανακάλυψη εσωτερικής δομής ή προτύπων. Κλασικά παραδείγματα αποτελούν οι αλγόριθμοι ομαδοποίησης (clustering), όπως ο k-Means, ο οποίος χωρίζει τα δεδομένα σε προκαθορισμένο αριθμό ομάδων ελαχιστοποιώντας την ενδο-ομαδική διασπορά. Επιπλέον, χρησιμοποιούνται μέθοδοι ιεραρχικής ομαδοποίησης (Hierarchical Clustering), οι οποίες κατασκευάζουν δενδρικές δομές συστάδων και επιτρέπουν την ανάλυση των δεδομένων σε διαφορετικά επίπεδα λεπτομέρειας.

Η **ενισχυτική μάθηση** διαφοροποιείται ουσιαστικά από τις προηγούμενες κατηγορίες, καθώς βασίζεται στη συνεχή αλληλεπίδραση ενός πράκτορα με το περιβάλλον του. Ο πράκτορας λαμβάνει αποφάσεις, αξιολογείται μέσω μιας συνάρτησης ανταμοιβής και μαθαίνει βελτιστοποιώντας τη μακροπρόθεσμη απόδοσή του. Χαρακτηριστικά παραδείγματα αποτελούν η μέθοδος Q-Learning και οι τεχνικές Policy Gradient, ενώ σε πιο σύνθετα σενάρια χρησιμοποιούνται προσεγγίσεις όπως τα Deep Q-Networks (DQN), τα οποία συνδυάζουν ενισχυτική μάθηση με νευρωνικά δίκτυα.

Η παρουσίαση των παραπάνω κατηγοριών και αλγορίθμων παρέχει το αναγκαίο θεωρητικό υπόβαθρο για την κατανόηση των απαιτήσεων που καλείται να υποστηρίξει μια σύγχρονη εφαρμογή νέφους για τη δημιουργία και εκτέλεση μοντέλων μηχανικής μάθησης. Στο πλαίσιο της παρούσας διπλωματικής εργασίας, η ανάγκη υποστήριξης ετερογενών αλγορίθμων και διαφορετικών ροών εκπαίδευσης και εκτέλεσης επηρέασε καθοριστικά τον σχεδιασμό της εφαρμογής, όπως αναλύεται στα επόμενα κεφάλαια.

2.2 Κύκλος Ζωής Μοντέλων Μηχανικής Μάθησης

Ο κύκλος ζωής ενός μοντέλου μηχανικής μάθησης αναφέρεται στο σύνολο των διαδοχικών και επαναληπτικών σταδίων που ακολουθούνται από τον αρχικό ορισμό ενός προβλήματος έως τη συνεχή βελτίωση του μοντέλου μετά τη χρήση του σε πραγματικά δεδομένα. Αποτελεί θεμελιώδη έννοια στη μηχανική μάθηση, καθώς περιγράφει τη μεθοδολογική διαδικασία ανάπτυξης, αξιολόγησης και συντήρησης ενός μοντέλου.

Ο κύκλος ζωής ενός μοντέλου μηχανικής μάθησης είναι κατ' ουσίαν κυκλικός, καθώς τα αποτελέσματα των τελικών σταδίων μπορούν να οδηγήσουν σε αναθεώρηση και επανάληψη προηγούμενων φάσεων, με σκοπό τη βελτίωση της απόδοσης και της γενικευσιμότητας του μοντέλου.

Ο κύκλος ζωής περιλαμβάνει διακριτά στάδια, τα οποία παρουσιάζονται αναλυτικά στις επόμενες παραγράφους.

Στάδια κύκλου ζωής μοντέλου μηχανικής μάθησης:

1. Ορισμός προβλήματος και ανάλυση απαιτήσεων

Στο αρχικό στάδιο προσδιορίζεται το πρόβλημα που πρόκειται να επιλυθεί και καθορίζονται οι στόχοι της ανάλυσης. Παράλληλα, ορίζονται οι απαιτήσεις ως προς την ακρίβεια, την απόδοση και τους περιορισμούς του μοντέλου, καθώς και ο τύπος της μηχανικής μάθησης που θα εφαρμοστεί, όπως ταξινόμηση, παλινδρόμηση ή ομαδοποίηση.

2. Συλλογή και κατανόηση δεδομένων

Στο στάδιο αυτό πραγματοποιείται η συλλογή των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση και αξιολόγηση του μοντέλου. Επιπλέον, γίνεται διερευνητική ανάλυση των δεδομένων, με στόχο την κατανόηση της δομής, της ποιότητας και των βασικών χαρακτηριστικών τους.

3. Προετοιμασία δεδομένων και μηχανική χαρακτηριστικών

Τα δεδομένα υφίστανται διαδικασίες προεπεξεργασίας, όπως καθαρισμό, μετασχηματισμό και κανονικοποίηση. Παράλληλα, εφαρμόζονται τεχνικές μηχανικής χαρακτηριστικών (feature engineering), με σκοπό τη βελτίωση της πληροφορίας που παρέχεται στο μοντέλο και την ενίσχυση της απόδοσής του.

4. Επιλογή αλγορίθμου και εκπαίδευση μοντέλου

Στο στάδιο αυτό επιλέγεται ο κατάλληλος αλγόριθμος μηχανικής μάθησης και πραγματοποιείται η εκπαίδευση του μοντέλου με βάση τα προετοιμασμένα δεδομένα. Το

αποτέλεσμα της διαδικασίας είναι ένα εκπαιδευμένο μοντέλο, το οποίο ενσωματώνει τα πρότυπα και τις συσχετίσεις που εντοπίζονται στα δεδομένα.

5. Αξιολόγηση και επικύρωση μοντέλου

Το εκπαιδευμένο μοντέλο αξιολογείται με τη χρήση κατάλληλων μετρικών απόδοσης, προκειμένου να εκτιμηθεί η ποιότητά του και η ικανότητά του να γενικεύει σε νέα, άγνωστα δεδομένα. Σε περίπτωση μη ικανοποιητικών αποτελεσμάτων, προηγούμενα στάδια του κύκλου ζωής επαναλαμβάνονται.

6. Διάθεση μοντέλου (Model Deployment)

Μετά την επιτυχή αξιολόγηση, το μοντέλο καθίσταται διαθέσιμο για χρήση, ώστε να μπορεί να παράγει προβλέψεις ή αποφάσεις σε πραγματικές συνθήκες λειτουργίας. Το μοντέλο αυτό μπορεί να ενσωματωθεί σε εφαρμογές ώστε να ενισχύσει την λειτουργικότητά τους.

7. Παρακολούθηση και βελτίωση

Στο τελικό στάδιο παρακολουθείται η απόδοση του μοντέλου κατά τη διάρκεια της χρήσης του. Η εμφάνιση νέων δεδομένων ή η μεταβολή των αρχικών συνθηκών μπορεί να οδηγήσει στην ανάγκη επανεκπαίδευσης ή βελτιστοποίησης του μοντέλου, ολοκληρώνοντας έτσι τον κύκλο ζωής και οδηγώντας σε έναν νέο επαναληπτικό κύκλο.

2.3 Υπολογιστικό Νέφος

Σύμφωνα με τον επίσημο ορισμό του National Institute of Standards and Technology (NIST), το υπολογιστικό νέφος (Cloud Computing) αποτελεί ένα μοντέλο που επιτρέπει την καθολική, εύκολη και κατ' απαίτηση πρόσβαση μέσω δικτύου σε έναν κοινόχρηστο χώρο διαμορφώσιμων υπολογιστικών πόρων, όπως δίκτυα, διακομιστές, αποθηκευτικά συστήματα, εφαρμογές και υπηρεσίες. Οι πόροι αυτοί μπορούν να διατεθούν και να αποδεσμευτούν γρήγορα με ελάχιστη διαχειριστική προσπάθεια ή αλληλεπίδραση με τον πάροχο.

Ο ορισμός αυτός αναδεικνύει το υπολογιστικό νέφος ως μια θεμελιώδη τεχνολογική προσέγγιση για την παροχή υπολογιστικών υπηρεσιών, διαφοροποιούμενη από τις παραδοσιακές τοπικές υποδομές.

2.3.1 Βασικά Χαρακτηριστικά και Πλεονεκτήματα του Υπολογιστικού Νέφους

Το υπολογιστικό νέφος χαρακτηρίζεται από πέντε βασικά γνωρίσματα, όπως ορίζονται από το NIST:

- Παροχή πόρων κατ' απαίτηση (On-demand self-service): Ο χρήστης μπορεί να δεσμεύσει υπολογιστικούς πόρους αυτόματα, χωρίς να απαιτείται ανθρώπινη παρέμβαση από τον πάροχο.
- Ευρεία πρόσβαση μέσω δικτύου (Broad network access): Οι υπηρεσίες είναι προσβάσιμες μέσω δικτύου από ετερογενείς συσκευές, όπως κινητά τηλέφωνα, φορητούς υπολογιστές και σταθμούς εργασίας.
- Συγκέντρωση πόρων (Resource pooling): Οι υπολογιστικοί πόροι του παρόχου συγκεντρώνονται σε κοινόχρηστη δεξαμενή και κατανέμονται δυναμικά μεταξύ πολλαπλών χρηστών ανάλογα με τη ζήτηση.
- Ταχεία ελαστικότητα (Rapid elasticity): Οι πόροι μπορούν να αυξομειώνονται αυτόματα και άμεσα, ώστε να ανταποκρίνονται στις εκάστοτε ανάγκες του φόρτου εργασίας.
- Μετρήσιμη υπηρεσία (Measured service): Η χρήση πόρων παρακολουθείται, ελέγχεται και αναφέρεται με διαφάνεια, επιτρέποντας χρέωση βάσει πραγματικής κατανάλωσης.

Τα χαρακτηριστικά αυτά συνεπάγονται σημαντικά πλεονεκτήματα τόσο για τον χρήστη όσο και για τον πάροχο υπηρεσιών. Από την πλευρά του χρήστη, μειώνεται το κόστος επένδυσης σε υποδομή, αυξάνεται η ευελιξία και διευκολύνεται η ανάπτυξη και η κλιμάκωση εφαρμογών. Από την πλευρά του παρόχου, επιτυγχάνεται αποδοτικότερη αξιοποίηση των πόρων και βελτιστοποίηση της λειτουργίας των υποδομών.

2.3.2 Μοντέλα Παροχής Υπηρεσιών Υπολογιστικού Νέφους

Το υπολογιστικό νέφος υλοποιείται μέσω τριών βασικών μοντέλων παροχής υπηρεσιών, όπως φαίνεται παρακάτω:

1. Το μοντέλο *Infrastructure as a Service (IaaS)* παρέχει βασικούς υπολογιστικούς πόρους, όπως εικονικούς διακομιστές, αποθηκευτικά συστήματα και δικτυακή υποδομή. Ο χρήστης διαχειρίζεται το λειτουργικό σύστημα και τις εφαρμογές, ενώ η φυσική υποδομή παραμένει υπό την ευθύνη του παρόχου. Το IaaS προσφέρει υψηλή ευελιξία και δυνατότητα προσαρμογής των πόρων με ευέλικτο μοντέλο χρέωσης με βάση την χρήση. Ενδεικτικές υπηρεσίες αποτελούν το Amazon EC2 και το Microsoft Azure Virtual Machines.

2. Το μοντέλο Platform as a Service (PaaS) προσφέρει ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών, απαλλάσσοντας τον χρήστη από τη διαχείριση της υποδομής, του λειτουργικού συστήματος και του περιβάλλοντος ανάπτυξης/εκτέλεσης. Η προσέγγιση αυτή μειώνει την πολυπλοκότητα, επιταχύνει τον κύκλο ανάπτυξης λογισμικού και επιτρέπει την offline εργασία ενώ υποστηρίζει και κλιμάκωση των προσφερόμενων περιβαλλόντων. Παραδείγματα υπηρεσιών PaaS είναι το Google App Engine και το Heroku.
3. Το μοντέλο Software as a Service (SaaS) παρέχει πλήρως λειτουργικές εφαρμογές μέσω διαδικτύου, χωρίς την ανάγκη τοπικής εγκατάστασης ή συντήρησης από τον χρήστη. Τα πλεονεκτήματα του SaaS περιλαμβάνουν την άμεση πρόσβαση, την ευκολία χρήσης και τη μείωση του λειτουργικού κόστους καθώς και τη σταθερή απόδοση λόγω της επίτευξης δυναμικής ελαστικότητας των εφαρμογών μέσω χρήσης κατάλληλων τεχνολογιών & υπηρεσιών νέφους. Χαρακτηριστικά παραδείγματα αποτελούν το Google Docs και το Microsoft Office 365.

2.3.3 Τεχνολογίες Απομόνωσης και Διαχείρισης Υπολογιστικών Πόρων

Η σύγχρονη αρχιτεκτονική υπολογιστικού νέφους βασίζεται σε τεχνολογίες που επιτρέπουν την απομόνωση, την αποδοτική αξιοποίηση και τη δυναμική διαχείριση των υπολογιστικών πόρων. Η ανάγκη για ευελιξία, κλιμακωσιμότητα και υψηλή διαθεσιμότητα οδήγησε στην ανάπτυξη μηχανισμών που επιτρέπουν τη λειτουργία πολλαπλών περιβαλλόντων εκτέλεσης πάνω στην ίδια φυσική υποδομή, διατηρώντας ταυτόχρονα λογική απομόνωση και ανεξαρτησία μεταξύ τους.

Οι βασικότερες τεχνολογικές προσεγγίσεις προς αυτή την κατεύθυνση είναι η εικονικοποίηση και η δοχειοποίηση. Η πρώτη εστιάζει στη δημιουργία πλήρως απομονωμένων εικονικών μηχανών μέσω επιπέδου διαχείρισης υλικού, ενώ η δεύτερη υλοποιεί απομόνωση σε επίπεδο λειτουργικού συστήματος, επιτρέποντας ελαφρύτερα και ταχύτερα περιβάλλοντα εκτέλεσης. Επιπλέον, η αυξανόμενη πολυπλοκότητα των σύγχρονων κατανεμημένων εφαρμογών κατέστησε αναγκαία την ανάπτυξη μηχανισμών ενορχήστρωσης, οι οποίοι αυτοματοποιούν τη διαχείριση, την κλιμάκωση και την ανθεκτικότητα των περιβαλλόντων αυτών.

Στις ακόλουθες υποενότητες παρουσιάζονται αναλυτικά οι έννοιες της εικονικοποίησης καθώς και της δοχειοποίησης σε συνδυασμό με τις τεχνολογίες ενορχήστρωσης.

2.3.3.1 Εικονικοποίηση

Η εικονικοποίηση αποτελεί βασική τεχνολογία του υπολογιστικού νέφους και επιτρέπει την αποδοτική αξιοποίηση των φυσικών πόρων. Μέσω της εικονικοποίησης, πολλαπλά απομονωμένα περιβάλλοντα εκτέλεσης μπορούν να λειτουργούν πάνω στο ίδιο φυσικό υλικό.

Διακρίνονται δύο βασικές μορφές εικονικοποίησης: η εικονικοποίηση σε επίπεδο υλικού, η οποία υλοποιείται μέσω hypervisors και δημιουργεί εικονικές μηχανές με πλήρες λειτουργικό σύστημα, και η εικονικοποίηση σε επίπεδο λειτουργικού συστήματος, όπου πολλαπλά περιβάλλοντα μοιράζονται τον ίδιο πυρήνα του λειτουργικού συστήματος. Η δεύτερη προσέγγιση χαρακτηρίζεται από μικρότερο κόστος πόρων και ταχύτερη εκκίνηση λόγω του μειωμένου footprint των σχετικών περιβαλλόντων της (των επονομαζόμενων δοχείων).

2.3.3.2 Δοχειοποίηση και Ενορχήστρωση

Η δοχειοποίηση (containerization) αποτελεί εξέλιξη της εικονικοποίησης σε επίπεδο λειτουργικού συστήματος και επιτρέπει τη συσκευασία εφαρμογών μαζί με τις εξαρτήσεις τους σε ελαφριά και απομονωμένα δοχεία (containers). Η τεχνολογία Docker αποτελεί την επικρατέστερη υλοποίηση δοχειοποίησης, προσφέροντας φορητότητα, συνέπεια μεταξύ περιβαλλόντων και ταχύτερη ανάπτυξη εφαρμογών.

Παρά τα πλεονεκτήματα της δοχειοποίησης, η χρήση μεμονωμένων δοχείων παρουσιάζει περιορισμούς ως προς την κλιμάκωση, την ανθεκτικότητα και τη διαχείριση σύνθετων συστημάτων. Οι περιορισμοί αυτοί οδήγησαν στην ανάγκη ενορχήστρωσης δοχείων.

Η ενορχήστρωση υλοποιείται μέσω τεχνολογιών όπως το Docker Swarm και το Kubernetes, οι οποίες παρέχουν μηχανισμούς αυτοματοποιημένης διαχείρισης, κατανομής φόρτου, παρακολούθησης και αυτοθεραπείας (self-healing) εφαρμογών. Μεταξύ αυτών, το Kubernetes έχει επικρατήσει ως de facto πρότυπο ενορχήστρωσης, προσφέροντας

αυξημένη επεκτασιμότητα, κλιμακωσιμότητα και αξιοπιστία, ιδιαίτερα σε περιβάλλοντα υπολογιστικού νέφους.

3 Σχετικές Εργασίες

Στον χώρο της μηχανικής μάθησης έχουν αναπτυχθεί πολυάριθμες πλατφόρμες και συστήματα που στοχεύουν στη διευκόλυνση της δημιουργίας, εκπαίδευσης και εκτέλεσης μοντέλων σε περιβάλλοντα υπολογιστικού νέφους. Οι περισσότερες από τις σύγχρονες λύσεις ενσωματώνουν μηχανισμούς αυτοματοποίησης, διαχείρισης πόρων και κλιμάκωσης, καλύπτοντας ανάγκες τόσο ερευνητικών όσο και επιχειρησιακών εφαρμογών.

Οι εμπορικές πλατφόρμες μηχανικής μάθησης προσφέρουν ολοκληρωμένα οικοσυστήματα, τα οποία συνδυάζουν εργαλεία ανάπτυξης, αποθήκευσης δεδομένων και εκτέλεσης υπολογιστικά απαιτητικών εργασιών. Ωστόσο, η στενή τους σύνδεση με συγκεκριμένους παρόχους υπολογιστικού νέφους συχνά περιορίζει τη φορητότητα και την ευελιξία των λύσεων αυτών, καθώς και τη δυνατότητα πειραματισμού με ετερογενείς αλγορίθμους και περιβάλλοντα εκτέλεσης.

Η παρούσα ενότητα εξετάζει αντιπροσωπευτικές πλατφόρμες και συστήματα που σχετίζονται με τη δημιουργία, εκπαίδευση και εκτέλεση μοντέλων μηχανικής μάθησης. Στόχος της ανάλυσης είναι η παρουσίαση των βασικών χαρακτηριστικών κάθε λύσης, καθώς και η ανάδειξη των πλεονεκτημάτων και των περιορισμών τους, προκειμένου να διαμορφωθεί ένα σαφές πλαίσιο σύγκρισης με το προτεινόμενο σύστημα.

Στις υποενότητες που ακολουθούν, κάθε πλατφόρμα αναλύεται ξεχωριστά, με έμφαση στη λειτουργικότητα που παρέχει και στη φιλοσοφία σχεδιασμού της, θέτοντας τις βάσεις για τη συνολική συγκριτική αποτίμηση που πραγματοποιείται στην Ενότητα 3.5.

3.1 Amazon SageMaker

Το Amazon SageMaker αποτελεί μία από τις πλέον διαδεδομένες εμπορικές πλατφόρμες μηχανικής μάθησης στο υπολογιστικό νέφος. Παρέχει ένα ολοκληρωμένο σύνολο εργαλείων για την προετοιμασία δεδομένων, την εκπαίδευση μοντέλων, την αξιολόγηση και την ανάπτυξή τους σε περιβάλλοντα παραγωγής. Η πλατφόρμα υποστηρίζει αυτοματοποιημένες ροές εργασίας και δυναμική κλιμάκωση, αξιοποιώντας την υποδομή της Amazon Web Services (AWS) (δηλ. του υπολογιστικού νέφους της Amazon).

Παρά τα πλεονεκτήματά του, το SageMaker χαρακτηρίζεται από ισχυρή εξάρτηση από το οικοσύστημα της AWS, καθώς και από προκαθορισμένες ροές εκτέλεσης. Παράλληλα, η υποστήριξη προσαρμοσμένων αλγορίθμων απαιτεί συμμόρφωση με συγκεκριμένες συμβάσεις και βαθιά γνώση της πλατφόρμας. Σε αντίθεση, η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας εστιάζει στην ανεξαρτησία από τον πάροχο νέφους και στη δυνατότητα εκτέλεσης πλήρως προσαρμοσμένων αλγορίθμων σε απομονωμένα περιβάλλοντα, χωρίς περιορισμούς ως προς τη γλώσσα ή το περιβάλλον υλοποίησης.

3.2 Google Vertex AI

Το Google Vertex AI αποτελεί μία ολοκληρωμένη πλατφόρμα μηχανικής μάθησης που παρέχεται στο οικοσύστημα του Google Cloud και ενοποιεί λειτουργίες ανάπτυξης, εκπαίδευσης, αξιολόγησης και διάθεσης μοντέλων. Υποστηρίζει τόσο έτοιμες υπηρεσίες AutoML όσο και προσαρμοσμένα μοντέλα (custom training), επιτρέποντας την εκτέλεση κώδικα σε προκαθορισμένα ή προσαρμοσμένα containers. Παράλληλα, ενσωματώνει μηχανισμούς MLOps, pipeline orchestration και διαχείρισης πειραμάτων, με ιδιαίτερη έμφαση σε εφαρμογές μεγάλης κλίμακας και βαθιάς μάθησης.

Η πλατφόρμα υποστηρίζει πολυχρηστικά και επιχειρησιακά περιβάλλοντα μέσω μηχανισμών διαχείρισης ταυτοτήτων και ρόλων (IAM), επιτρέποντας διακριτό έλεγχο πρόσβασης σε έργα και πόρους. Ωστόσο, η λειτουργία της είναι στενά συνδεδεμένη με το οικοσύστημα του Google Cloud, γεγονός που περιορίζει τη φορητότητα της λύσης εκτός της συγκεκριμένης υποδομής. Επιπλέον, η αρχιτεκτονική της προσανατολίζεται κυρίως σε production-grade MLOps σενάρια, παρά σε ελαφριά, cloud-agnostic εκπαιδευτικά περιβάλλοντα πειραματισμού.

Αντιθέτως, η προτεινόμενη εφαρμογή της παρούσας εργασίας δίνει έμφαση στην ανεξαρτησία από πάροχο νέφους, στην εκτέλεση πλήρως ετερογενούς κώδικα χωρίς τεχνολογικούς περιορισμούς και στη διαφάνεια της διαδικασίας εκπαίδευσης σε ακαδημαϊκά και ερευνητικά περιβάλλοντα.

3.3 Microsoft Azure Machine Learning

Το Azure Machine Learning αποτελεί την αντίστοιχη πλατφόρμα μηχανικής μάθησης της Microsoft, προσφέροντας εργαλεία για την εκπαίδευση, την παρακολούθηση και την ανάπτυξη μοντέλων. Υποστηρίζει τόσο γραφικές διεπαφές όσο και προγραμματιστικές διεπαφές, διευκολύνοντας την ενσωμάτωση με τις υπόλοιπες υπηρεσίες του Azure (δηλ. του υπολογιστικού νέφους της Microsoft).

Παρόμοια με άλλες εμπορικές πλατφόρμες, το Azure Machine Learning βασίζεται σε κλειστό οικοσύστημα και σε προκαθορισμένα πρότυπα εκτέλεσης. Η εφαρμογή της παρούσας εργασίας διαφοροποιείται, καθώς επιτρέπει την εκτέλεση προσαρμοσμένων αλγορίθμων σε απομονωμένα δοχεία, με σαφή όρια εμπιστοσύνης (trust boundaries), χωρίς να απαιτείται δέσμευση σε συγκεκριμένη υποδομή ή πάροχο.

3.4 MLflow

Το MLflow είναι ένα ανοικτού κώδικα εργαλείο διαχείρισης πειραμάτων μηχανικής μάθησης, το οποίο χρησιμοποιείται κυρίως για την καταγραφή παραμέτρων, μετρικών και εκδόσεων μοντέλων. Αποτελεί δημοφιλή επιλογή σε ερευνητικά περιβάλλοντα και σε έργα πειραματικού χαρακτήρα.

Παρότι το MLflow προσφέρει σημαντική υποστήριξη στην παρακολούθηση πειραμάτων, δεν παρέχει πλήρη μηχανισμό εκτέλεσης και ενορχήστρωσης υπολογιστικών φορτίων, ούτε ενσωματωμένη διαχείριση πολυχρηστικών σεναρίων. Η προτεινόμενη εφαρμογή επεκτείνει αυτήν τη φιλοσοφία, ενσωματώνοντας μηχανισμούς ασύγχρονης εκτέλεσης, αποθήκευσης δεδομένων και απομονωμένης εκτέλεσης αλγορίθμων (Μηχανικής Μάθησης) μέσω Docker και Kubernetes.

3.5 Σύγκριση με την Προτεινόμενη Εφαρμογή

Για τη συνολική αποτίμηση των υφιστάμενων πλατφορμών σε σχέση με το προτεινόμενο σύστημα, κρίνεται απαραίτητη η χρήση συγκεκριμένων κριτηρίων σύγκρισης που αντανακλούν βασικές απαιτήσεις σύγχρονων εφαρμογών μηχανικής μάθησης σε περιβάλλοντα υπολογιστικού νέφους. Τα κριτήρια αυτά επιλέχθηκαν ώστε να καλύπτουν τόσο τεχνικές όσο και λειτουργικές πτυχές, με ιδιαίτερη έμφαση στη χρηστικότητα, την ευελιξία και την επεκτασιμότητα.

Ένα από τα βασικότερα κριτήρια είναι η υποστήριξη προσαρμοσμένων αλγορίθμων μηχανικής μάθησης. Η δυνατότητα εισαγωγής και εκτέλεσης αλγορίθμων που υλοποιούνται από τους ίδιους τους χρήστες θεωρείται κρίσιμη, ιδίως σε εκπαιδευτικά και ερευνητικά περιβάλλοντα, όπου απαιτείται πειραματισμός με νέες μεθόδους και υλοποιήσεις πέραν προκαθορισμένων βιβλιοθηκών.

Εξίσου σημαντικό κριτήριο αποτελεί η ανεξαρτησία από τον πάροχο υπολογιστικού νέφους (cloud agnosticism). Η απαίτηση αυτή σχετίζεται με την αποφυγή εγκλωβισμού σε συγκεκριμένα οικοσυστήματα και με τη δυνατότητα μεταφοράς ή αναπαραγωγής της εφαρμογής σε διαφορετικά περιβάλλοντα υποδομής, συμπεριλαμβανομένων ιδιωτικών ή υβριδικών νεφών.

Η ασύγχρονη εκτέλεση εργασιών αποτελεί επίσης βασική απαίτηση, καθώς η εκπαίδευση και εκτέλεση μοντέλων μηχανικής μάθησης είναι συχνά χρονοβόρες και υπολογιστικά απαιτητικές διεργασίες. Η ασύγχρονη υλοποίηση επιτρέπει την αποσύνδεση του αιτήματος του χρήστη από τη διάρκεια εκτέλεσης, βελτιώνοντας την απόκριση και τη συνολική εμπειρία χρήσης.

Τέλος, η υποστήριξη πολυχρηστικού περιβάλλοντος και η καταλληλότητα για εκπαιδευτική και ερευνητική χρήση αποτελούν κρίσιμους παράγοντες αξιολόγησης. Η ύπαρξη διακριτών ρόλων, ιδιοκτησίας πόρων και διαφάνειας στη διαδικασία εκπαίδευσης και εκτέλεσης είναι απαραίτητη για συνεργατικά σενάρια και ακαδημαϊκά πλαίσια.

Με βάση τα παραπάνω κριτήρια, ο Πίνακας 3.1 παρουσιάζει μια συνοπτική σύγκριση αντιπροσωπευτικών πλατφορμών μηχανικής μάθησης με το προτεινόμενο σύστημα.

Πίνακας 3.1 Συνοπτική Σύγκριση Παρόμοιων Συστημάτων

Σύστημα	**Προσαρμοσμένοι Αλγόριθμοι**	**Cloud Agnostic**	**Ασύγχρονη Εκτέλεση**	**Πολυχρηστικό**	**Εκπαιδευτική Στόχευση**
Amazon SageMaker	Περιορισμένη υποστήριξη	✗	✓	✓	✗
Google Vertex AI	Περιορισμένη υποστήριξη	✗	✓	✓	✗
Microsoft Azure ML	Περιορισμένη υποστήριξη	✗	✓	✓	✗
MLflow	✗	✓	✗	✗	✓

Προτεινόμενο Σύστημα	✓	✓	✓	✓	✓

Από τη συγκριτική αποτύπωση προκύπτει ότι οι εμπορικές πλατφόρμες, όπως οι Amazon SageMaker, Google Vertex AI και Microsoft Azure Machine Learning, προσφέρουν ισχυρές δυνατότητες σε επίπεδο υποδομής και κλιμάκωσης, ωστόσο είναι στενά συνδεδεμένες με τα αντίστοιχα οικοσυστήματα παρόχων και περιορίζουν την ευελιξία ως προς την πλήρη ενσωμάτωση ετερογενών αλγορίθμων. Από την πλευρά των εργαλείων ανοικτού κώδικα, το MLflow αποτελεί μία αξιόλογη λύση για την καταγραφή πειραμάτων και την εκδοσιοποίηση μοντέλων, προσφέροντας ανεξαρτησία από πάροχο νέφους και καταλληλότητα για ερευνητικά περιβάλλοντα. Ωστόσο, δεν παρέχει ενσωματωμένο μηχανισμό εκτέλεσης και ενορχήστρωσης υπολογιστικών φορτίων, ούτε υποστηρίζει πολυχρηστικά σενάρια με διακριτό έλεγχο πρόσβασης και διαμοιρασμό πόρων. Κατά συνέπεια, η χρήση του προϋποθέτει την ύπαρξη εξωτερικής υποδομής εκτέλεσης, η οποία πρέπει να διαχειριστεί ανεξάρτητα από τον χρήστη ή τον οργανισμό.

Αντίθετα, η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας εστιάζει στη δημιουργία ενός ευέλικτου και επεκτάσιμου πλαισίου, το οποίο δίνει έμφαση στην απομόνωση εκτέλεσης μέσω δοχειοποίησης, στη διαφάνεια των διαδικασιών και στην ανεξαρτησία από τον πάροχο νέφους. Η πλήρης παρακολούθηση της κατάστασης των εργασιών και η διάθεση αναλυτικών logs (καταχωρήσεων) καθιστούν το σύστημα ιδιαίτερα κατάλληλο για εκπαιδευτικά και ερευνητικά περιβάλλοντα, όπου η κατανόηση της διαδικασίας και η επαναληψιμότητα αποτελούν βασικές απαιτήσεις.

Παρότι το προτεινόμενο σύστημα δεν αποσκοπεί στην παρούσα φάση στην πλήρη αντικατάσταση εμπορικών λύσεων μεγάλης κλίμακας, η αρχιτεκτονική και οι σχεδιαστικές επιλογές του δημιουργούν τις βάσεις για μελλοντική εξέλιξη προς πιο ολοκληρωμένες και παραγωγικές κατευθύνσεις. Η προοπτική αυτή εξετάζεται στο Κεφάλαιο 6, στο πλαίσιο των μελλοντικών επεκτάσεων της εφαρμογής.

4 Ανάπτυξη Εφαρμογής

Το παρόν κεφάλαιο παρουσιάζει την ανάπτυξη της εφαρμογής *ThesisApp*, η οποία υλοποιεί μια cloud-native πλατφόρμα για τη δημιουργία, εκπαίδευση και εκτέλεση μοντέλων μηχανικής μάθησης. Για την ανάπτυξη της εφαρμογής υιοθετήθηκε το μοντέλο της επαναληπτικής ανάπτυξης (Iterative Development Model), σύμφωνα με το οποίο η διαδικασία ανάλυσης απαιτήσεων, σχεδιασμού, υλοποίησης και δοκιμής πραγματοποιείται σε διαδοχικούς κύκλους. Σε κάθε επανάληψη ενσωματώνονται νέες λειτουργίες ή βελτιώσεις, επιτρέποντας τη σταδιακή ωρίμανση του συστήματος και την προσαρμογή του σε νέες απαιτήσεις ή τεχνικές παρατηρήσεις που προκύπτουν κατά την πορεία της ανάπτυξης.

Η ανάλυση οργανώνεται με τρόπο που ακολουθεί αυτήν το μοντέλο αυτό, ξεκινώντας από την αποτύπωση των λειτουργικών και μη λειτουργικών απαιτήσεων και καταλήγοντας στην τεχνική υλοποίηση και την αξιολόγηση της ορθότητας μέσω δοκιμών. Συνεπώς, σκοπός του παρόντος κεφαλαίου είναι να αναλύσει τα αποτελέσματα κάθε φάσης ανάπτυξης σε ξεχωριστές ενότητες, τεκμηριώνοντας συστηματικά τη μετάβαση από τις απαιτήσεις στις σχεδιαστικές και υλοποιητικές επιλογές, οι οποίες εξασφαλίζουν επεκτασιμότητα, ασφάλεια και συντηρησιμότητα.

4.1 Απαιτήσεις Συστήματος

Οι απαιτήσεις συστήματος προσδιορίζουν τις λειτουργίες, τα ποιοτικά χαρακτηριστικά και τους περιορισμούς που διέπουν τη σχεδίαση και την υλοποίηση της εφαρμογής. Ειδικότερα, στην παρούσα ενότητα παρουσιάζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις, καθώς και οι τεχνολογικοί περιορισμοί και εξαρτήσεις που λήφθηκαν υπόψη κατά την ανάπτυξη του συστήματος. Η ανάλυση αυτή αποτέλεσε τη βάση για τις αρχιτεκτονικές & σχεδιαστικές επιλογές και τις αποφάσεις υλοποίησης της εφαρμογής/συστήματος που παρουσιάζονται στις επόμενες ενότητες.

4.1.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τις δυνατότητες και τις υπηρεσίες που οφείλει να παρέχει η εφαρμογή στους χρήστες της, ανεξάρτητα από τον τρόπο υλοποίησής της. Οι απαιτήσεις αυτές αποτυπώνουν τις βασικές περιπτώσεις χρήσης (use cases) του συστήματος και ομαδοποιούνται ανά λειτουργική περιοχή, λαμβάνοντας υπόψη τους διαφορετικούς ρόλους χρηστών.

I. Διαχείριση Χρηστών

Εγγραφή Χρηστών. Η εφαρμογή θα πρέπει να υποστηρίζει την εγγραφή νέων χρηστών με την παροχή των εξής πεδίων: ονοματεπώνυμο, email, κωδικός πρόσβασης (εις διπλούν), ηλικία, επάγγελμα και χώρα διαμονής, εκ των οποίων τα τρία πρώτα είναι υποχρεωτικά. Ο κωδικός πρόσβασης θα πρέπει να ακολουθεί κατάλληλο μοτίβο ασφαλείας (γράμματα, ψηφία, ειδικοί χαρακτήρες, ελάχιστο μήκος 8 χαρακτήρων) και να παρέχεται πανομοιότυπος και στα δύο πεδία.

Αυθεντικοποίηση. Η αυθεντικοποίηση θα πραγματοποιείται μέσω ονόματος χρήστη και κωδικού, με την έκδοση μάρκας (token) προσωρινής διάρκειας για την πρόσβαση σε λειτουργίες που απαιτούν εξουσιοδότηση. *Άλλες Λειτουργίες*. Θα υποστηρίζεται επίσης η τροποποίηση στοιχείων χρήστη (από τον ίδιο ή τον διαχειριστή, εξαιρουμένου του κωδικού), η αλλαγή κωδικού (με παροχή παλαιού και νέου κωδικού εις διπλούν), η επαναφορά κωδικού μέσω αποστολής συνδέσμου στο email του χρήστη, καθώς και η διαγραφή λογαριασμού τόσο από τον ίδιο τον χρήστη όσο και από τον διαχειριστή.

II. Διαχείριση Συνόλων Δεδομένων

Το σύστημα θα πρέπει να παρέχει πλήρη υποστήριξη για τη διαχείριση συνόλων δεδομένων, συμπεριλαμβανομένης της μεταφόρτωσης αρχείων σε μορφή CSV, ARFF ή XLSX. Θα υποστηρίζεται η αυτόματη ανίχνευση τύπων στηλών, η ρύθμιση παραμέτρων του συνόλου δεδομένων (επιλογή βασικών χαρακτηριστικών και στήλης-στόχου), η προεπισκόπηση περιεχομένου, η δημιουργία αντιγράφων, η κοινοποίηση σε άλλους χρήστες και η διαγραφή συνόλων δεδομένων.

III. Διαχείριση Εκπαιδεύσεων

Η εφαρμογή θα πρέπει να επιτρέπει την εκπαίδευση μοντέλων μηχανικής μάθησης μέσω ενσωματωμένων ή προσαρμοσμένων αλγορίθμων. Ο χρήστης θα παρέχει ένα σύνολο δεδομένων κατάλληλης μορφοποίησης (CSV, ARFF ή XLSX) και θα προσδιορίζει ποιες στήλες αποτελούν τα βασικά χαρακτηριστικά και ποια είναι η στήλη πρόβλεψης — προκαθορισμένα, όλες οι στήλες εκτός της τελευταίας χρησιμοποιούνται ως

χαρακτηριστικά και η τελευταία ως πρόβλεψης. Ο χρήστης θα μπορεί να διαμορφώνει τις παραμέτρους (υπερπαραμέτρους) του αλγόριθμου ή να χρησιμοποιεί τις προκαθορισμένες τιμές.

Δεδομένου ότι η εκπαίδευση μπορεί να απαιτήσει σημαντικό χρόνο, η εκτέλεση θα είναι ασύγχρονη: εφόσον η αίτηση είναι έγκυρη, θα επιστρέφεται ένα αναγνωριστικό εκπαίδευσης με το οποίο ο χρήστης θα μπορεί να παρακολουθεί την κατάστασή της. Οι δυνατές καταστάσεις θα είναι: προς εκκίνηση, εκκινημένη, ολοκληρωμένη και αποτυχημένη. Για τις δύο τελευταίες καταστάσεις (που είναι τελικές) θα επιστρέφονται λεπτομέρειες εκτέλεσης (αποτελέσματα κονσόλας) καθώς και η ημερομηνία ολοκλήρωσης ή αποτυχίας.

Θα υποστηρίζονται επίσης: η εμφάνιση λίστας εκπαιδεύσεων με δυνατότητα φιλτραρίσματος (βάσει εύρους ημερομηνιών ή αλγορίθμου), η εκφόρτωση του παραγόμενου μοντέλου, το σταμάτημα εκπαίδευσης σε εξέλιξη, η διαγραφή εκπαίδευσης (μαζί με τα αποτελέσματά της) και η επανεκπαίδευση βάσει υπάρχουσας εκπαίδευσης με δυνατότητα τροποποίησης παραμέτρων, η οποία θα οδηγεί στη δημιουργία νέας εγγραφής εκπαίδευσης.

IV. Υποστήριξη Προσαρμοσμένων Αλγορίθμων

Η εφαρμογή, εκτός από τους ενσωματωμένους αλγόριθμους μηχανικής μάθησης, θα πρέπει να επιτρέπει στον χρήστη να εισάγει και να χρησιμοποιεί τους δικούς του αλγόριθμους. Ο κώδικας του αλγόριθμου θα πρέπει να είναι πακεταρισμένος με τη μορφή εικόνας δοχείου (Docker image), η οποία θα μεταφορτώνεται είτε ως αρχείο TAR είτε μέσω URL δημοσίου αποθετηρίου (Docker Hub). Κατά την εισαγωγή, ο χρήστης θα παρέχει κατάλληλα μεταδεδομένα: όνομα, κειμενική περιγραφή, έκδοση, λέξεις-κλειδιά, είδος πρόσβασης (δημόσιος ή ιδιωτικός) και παραμέτρους διαμόρφωσης με κατάλληλο εύρος τιμών.

Η εκπαίδευση σε αυτή την περίπτωση θα πραγματοποιείται μέσω αντιστοίχισης του φακέλου δεδομένων σε σημείο προσάρτησης (mount point) μέσα στο δοχείο του αλγόριθμου, ενώ τα αποτελέσματα θα αντλούνται από αντίστοιχο σημείο εξόδου. Η χρήση δοχείων διασφαλίζει την απομονωμένη εκτέλεση και την αντιμετώπιση ζητημάτων ασφαλείας. Θα υποστηρίζεται επίσης η τροποποίηση μεταδεδομένων και περιεχομένου αλγόριθμου, η διαγραφή του (μαζί με τις συσχετισμένες εκπαιδεύσεις και μοντέλα), η εμφάνιση λεπτομερειών, η αναζήτηση (απλή και εξεζητημένη) καθώς και η κοινοποίηση σε άλλους χρήστες. Οι ιδιωτικοί αλγόριθμοι θα εμφανίζονται, αναζητούνται και χρησιμοποιούνται αποκλειστικά από τον κάτοχό τους.

V. Διαχείριση Μοντέλων Μηχανικής Μάθησης

Ο χρήστης θα μπορεί να διατηρεί το αποτέλεσμα μιας επιτυχημένης εκπαίδευσης με τη μορφή μοντέλου μηχανικής μάθησης, παρέχοντας κατάλληλα μεταδεδομένα: όνομα μοντέλου, λειτουργική κατηγορία, λέξεις-κλειδιά, κειμενική περιγραφή, περιγραφή συνόλου δεδομένων και είδος πρόσβασης (δημόσιο ή ιδιωτικό). Το σύστημα θα παράγει αυτόματα αναγνωριστικό μοντέλου και θα καταχωρεί τις ημερομηνίες εκπαίδευσης και δημιουργίας.

Θα υποστηρίζεται η τροποποίηση μεταδεδομένων και περιεχομένου μοντέλου (μέσω αντιστοίχισης με νέα εκπαίδευση), η εμφάνιση λεπτομερειών και μετρικών αξιολόγησης, η εκφόρτωση αρχείου μοντέλου (μόνο από τον κάτοχο), η διαγραφή (μόνο από τον κάτοχο, χωρίς διαγραφή της υποκείμενης εκπαίδευσης) και η κοινοποίηση σε άλλους χρήστες.

Για την αναζήτηση μοντέλων θα υποστηρίζονται δύο τρόποι: η απλή αναζήτηση μέσω λέξεων-κλειδιών που ταιριάζουν με οποιοδήποτε μεταδεδομένο, και η εξεζητημένη αναζήτηση με συγκεκριμένες τιμές ανά μεταδεδομένο, σημασιολογία λογικού ΚΑΙ ή Ή, εύρος ημερομηνιών και επιλογή κατηγοριών. Τα ιδιωτικά μοντέλα θα εμφανίζονται αποκλειστικά στον κάτοχό τους.

VI. Εκτέλεση Προβλέψεων

Η εφαρμογή θα επιτρέπει την εκτέλεση προβλέψεων μέσω εκπαιδευμένων μοντέλων, στον κάτοχο σε κάθε περίπτωση και σε λοιπούς χρήστες εφόσον το μοντέλο είναι δημόσιο. Ο χρήστης θα μεταφορτώνει ένα σύνολο δεδομένων με την ίδια μορφή χαρακτηριστικών που χρησιμοποιήθηκε κατά την εκπαίδευση. Η εκτέλεση θα πραγματοποιείται ασύγχρονα σε απομονωμένο περιβάλλον δοχείου, με δυνατότητα παρακολούθησης κατάστασης. Τα αποτελέσματα πρόβλεψης θα είναι διαθέσιμα για εκφόρτωση σε μορφή αρχείου.

Θα υποστηρίζεται πλήρης διαχείριση εκτελέσεων: επανεκτέλεση μοντέλου (ιδιαιτέρως χρήσιμη μετά από επανεκπαίδευση), διαγραφή εκτέλεσης μαζί με τα δεδομένα εισόδου και εξόδου, αναζήτηση εκτελέσεων βάσει χρονικού διαστήματος, και εμφάνιση μεταδεδομένων εκτέλεσης (ημερομηνία, μοντέλο αναφοράς, κατάσταση) με δυνατότητα εκφόρτωσης αποτελεσμάτων.

VII. Διαχείριση Κατηγοριών

Η εφαρμογή θα υποστηρίζει την ιεραρχική κατηγοριοποίηση μοντέλων μηχανικής μάθησης και συνόλων δεδομένων, καθώς και τη συνεργατική διαχείρισή της. Οι αυθεντικοποιημένοι χρήστες θα μπορούν να προτείνουν αλλαγές (δημιουργία νέας κατηγορίας, τροποποίηση ή διαγραφή υπάρχουσας), ενώ ο διαχειριστής θα μπορεί να τις αποδέχεται, να τις απορρίπτει ή να πραγματοποιεί άμεσα αλλαγές στην ιεραρχία. Κάθε κατηγορία θα περιλαμβάνει όνομα και προαιρετική περιγραφή, και θα μπορεί να εντάσσεται ως υποκατηγορία υπάρχουσας ή

στο ανώτατο επίπεδο. Σε περίπτωση διαγραφής κατηγορίας που συνδέεται μοναδικά με μοντέλο, το μοντέλο θα μεταφέρεται στην υπέρτατη κατηγορία.

4.1.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις καθορίζουν τα ποιοτικά χαρακτηριστικά του συστήματος και τους περιορισμούς που επηρεάζουν τη συνολική συμπεριφορά και απόδοσή του. Οι απαιτήσεις αυτές αφορούν, μεταξύ άλλων, την ασφάλεια, την απόδοση, την κλιμάκωση, την αξιοπιστία, τη φορητότητα, τη διαλειτουργικότητα και τη συντηρησιμότητα της εφαρμογής.

Ιδιαίτερη έμφαση δίνεται στην ασφάλεια, όπου απαιτείται η χρήση μηχανισμών αυθεντικοποίησης και εξουσιοδότησης βάσει JWT tokens με ασύμμετρη κρυπτογράφηση RSA και ρόλων χρηστών (RBAC), ο κατακερματισμός κωδικών πρόσβασης μέσω του αλγόριθμου Argon2id, η δυνατότητα ανάκλησης tokens κατά την αποσύνδεση, η απομόνωση των εργασιών μηχανικής μάθησης σε ξεχωριστά περιβάλλοντα εκτέλεσης χωρίς δικαιώματα root και χωρίς πρόσβαση στο δίκτυο του host, καθώς και η χρήση κρυπτογραφημένης επικοινωνίας (TLS) μέσω reverse proxy ή Ingress Controller για την προστασία των δεδομένων κατά τη μεταφορά.

Η φορητότητα του συστήματος θα πρέπει να διασφαλιστεί μέσω της δοχειοποίησης (containerization) όλων των στοιχείων της εφαρμογής σε Docker images. Κάθε υπηρεσία — backend, frontend, εκτελεστής αλγόριθμων (Weka Runner) — θα πρέπει να πακετάρεται σε ανεξάρτητη εικόνα δοχείου, εξαλείφοντας τις εξαρτήσεις από το περιβάλλον εκτέλεσης. Η χρήση πολυσταδιακής κατασκευής (multi-stage build) στα production images θα μπορεί να μειώσει το μέγεθος και την επιφάνεια επίθεσης. Η ενορχήστρωση μέσω Kubernetes θα παρέχει επιπλέον δυνατότητες: αυτόματη ανάνηψη αποτυχημένων pods μέσω ελέγχων υγείας(liveness/readiness probes), δηλωτική διαχείριση πόρων (resource requests/limits), μόνιμη αποθήκευση μέσω PersistentVolumeClaims, διαχείριση ευαίσθητων δεδομένων μέσω Kubernetes Secrets, ανακάλυψη υπηρεσιών μέσω εσωτερικού DNS, και έλεγχο πρόσβασης στο επίπεδο cluster μέσω RBAC (ServiceAccount με εξειδικευμένο Role για διαχείριση Jobs). Το σύστημα δεν πρέπει να εξαρτάται από συγκεκριμένο πάροχο cloud, καθώς τα manifests είναι συμβατά με οποιοδήποτε Kubernetes cluster.

Παράλληλα, η εφαρμογή θα πρέπει να σχεδιαστεί ώστε να υποστηρίζει οριζόντια κλιμάκωση και ταυτόχρονη εξυπηρέτηση πολλαπλών χρηστών μέσω πολλαπλών replicas, διατηρώντας αποδεκτά επίπεδα απόδοσης μέσω ασύγχρονης εκτέλεσης χρονοβόρων

εργασιών. Η αξιοπιστία θα πρέπει να ενισχυθεί μέσω ελέγχων υγείας σε κάθε υπηρεσία, διαχείρισης συναλλαγών (@Transactional) και αυτόματου καθαρισμού πόρων (αυτοδιαγραφή ολοκληρωμένων Jobs, εκκαθάριση προσωρινών αρχείων). Τέλος, η διαλειτουργικότητα θα πρέπει να διασφαλιστεί μέσω RESTful API επιπέδου τουλάχιστον 2 στο μοντέλο ωριμότητας Richardson, υποστήριξης πολλαπλών μορφών δεδομένων (CSV, XLSX, ARFF, JSON) και της δυνατότητας εκτέλεσης αλγόριθμων σε οποιαδήποτε γλώσσα προγραμματισμού μέσω του μηχανισμού BYOC (Bring Your OwnContainer).

Η συντηρησιμότητα του συστήματος θα πρέπει να διασφαλιστεί μέσω σαφούς διαχωρισμού ευθυνών σε λογικά επίπεδα (presentation, business logic, data access), χρήσης αφαιρετικών μηχανισμών που επιτρέπουν την προσθήκη νέων αλγορίθμων ή τύπων εργασιών χωρίς τροποποίηση του υπάρχοντος κώδικα, καθώς και αυτοματοποιημένης διαχείρισης σχήματος βάσης δεδομένων μέσω μηχανισμού migrations. Ο σχεδιασμός αυτός στοχεύει στη μείωση του κόστους μελλοντικών τροποποιήσεων και στη διευκόλυνση της επεκτασιμότητας του συστήματος.

4.1.3 Τεχνολογικοί Περιορισμοί και Εξαρτήσεις

Η ανάπτυξη και λειτουργία της εφαρμογής υπόκειται σε συγκεκριμένους τεχνολογικούς περιορισμούς και εξαρτήσεις, οι οποίοι καθορίστηκαν εξαρχής και απορρέουν από τη φύση του προβλήματος και τις αρχιτεκτονικές απαιτήσεις του συστήματος.

Ο βασικότερος περιορισμός αφορά την ανάγκη ύπαρξης περιβάλλοντος δοχειοποίησης και ενορχήστρωσης, καθώς η εκτέλεση εργασιών μηχανικής μάθησης πραγματοποιείται αποκλειστικά ως απομονωμένες εργασίες (jobs) σε containers. Η αρχιτεκτονική αυτή επιβάλλει ότι κάθε αλγόριθμος — είτε προκαθορισμένος είτε προσαρμοσμένος — πρέπει να είναι πακεταρισμένος σε εικόνα δοχείου (container image) και να ακολουθεί ένα συγκεκριμένο συμβόλαιο εισόδου/εξόδου ως προς τη μορφή και τη δομή των δεδομένων.

Επιπλέον, η εφαρμογή εξαρτάται από τα ακόλουθα, καθένα εκ των οποίων επιβάλλεται από συγκεκριμένες αρχιτεκτονικές και λειτουργικές ανάγκες:

- Σχεσιακή βάση δεδομένων για την αποθήκευση μεταδεδομένων (χρήστες, μοντέλα, εκπαιδεύσεις, κατηγορίες). Η ανάγκη για σύνθετες σχέσεις μεταξύ οντοτήτων (π.χ. εκπαίδευση → διαμόρφωση αλγορίθμου → παράμετροι) και η εξασφάλιση ακεραιότητας δεδομένων μέσω περιορισμών ξένων κλειδιών καθιστούν τη σχεσιακή βάση απαραίτητη.

- Αποθετήριο αντικειμένων (object storage) συμβατό με το πρότυπο S3 για την αποθήκευση αρχείων δεδομένων, μοντέλων και αποτελεσμάτων. Τα αρχεία αυτά είναι μη δομημένα και ποικίλου μεγέθους, γεγονός που καθιστά ακατάλληλη την αποθήκευσή τους σε σχεσιακή βάση, ενώ η συμβατότητα με S3 εξασφαλίζει φορητότητα σε διαφορετικούς παρόχους νέφους.
- Υπηρεσίες δικτύου (DNS, ingress routing) για τη δρομολόγηση αιτημάτων προς τις επιμέρους υπηρεσίες του συστήματος. Ο διαχωρισμός frontend, backend και object storage σε διακριτές υπηρεσίες απαιτεί κεντρικό σημείο δρομολόγησης για την ενιαία αντιμετώπιση των αιτημάτων.
- Υπηρεσία αποστολής ηλεκτρονικού ταχυδρομείου (SMTP) για λειτουργίες όπως η επαναφορά κωδικού πρόσβασης. Η λειτουργία αυτή αποτελεί βασική απαίτηση ασφαλείας σε πολυχρηστικά συστήματα, καθώς επιτρέπει την ταυτοποίηση του χρήστη μέσω εξωτερικού καναλιού.
- Επαρκείς υπολογιστικούς πόρους (CPU, μνήμη, αποθηκευτικός χώρος), καθώς η εκπαίδευση μοντέλων μηχανικής μάθησης μπορεί να είναι ιδιαίτερα απαιτητική, ειδικά για μεγάλα σύνολα δεδομένων ή σύνθετους αλγορίθμους. Η εκτέλεση κάθε εκπαίδευσης ως ξεχωριστό Kubernetes Job σημαίνει ότι πολλαπλές ταυτόχρονες εκπαιδεύσεις πολλαπλασιάζουν τις απαιτήσεις σε πόρους.

Τέλος, η ασύγχρονη φύση των εργασιών μηχανικής μάθησης επιβάλλει την ύπαρξη μηχανισμών παρακολούθησης κατάστασης, διαχείρισης αποτυχιών και ασφαλούς τερματισμού, ώστε να διασφαλίζεται η αξιοπιστία του συστήματος ακόμη και σε περιπτώσεις απρόβλεπτης διακοπής ή εξάντλησης πόρων.

4.2 Σχεδίαση Συστήματος

Η παρούσα ενότητα περιγράφει τη σχεδίαση του συστήματος σε διαδοχικά επίπεδα αφαίρεσης, ξεκινώντας από την υψηλού επιπέδου επισκόπηση του περιβάλλοντος λειτουργίας και καταλήγοντας στη λεπτομερή ανάλυση της αρχιτεκτονικής, των περιπτώσεων χρήσης, των ροών αλληλεπιδράσεων και της πληροφοριακής δομής του συστήματος. Στόχος της ενότητας είναι να καταδείξει πώς οι λειτουργικές και μη λειτουργικές απαιτήσεις που παρουσιάστηκαν στην προηγούμενη ενότητα μετασχηματίζονται σε συγκεκριμένες σχεδιαστικές επιλογές, οι οποίες υποστηρίζουν την κλιμακωσιμότητα, την ασφάλεια και τη συντηρησιμότητα του συστήματος.

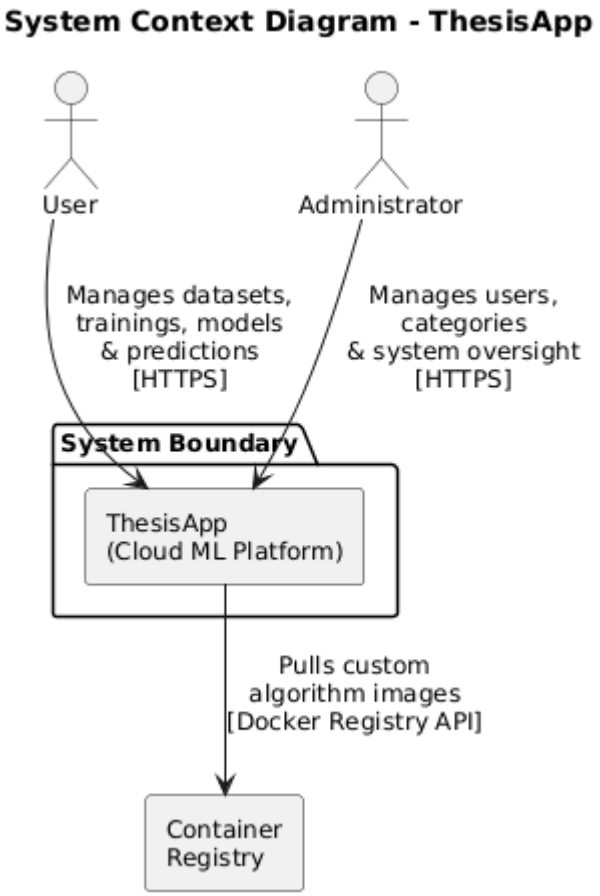
4.2.1 Διάγραμμα Περιβάλλοντος (System Context Diagram)

Το διάγραμμα περιβάλλοντος (δείτε Εικόνα 4.1) παρουσιάζει την εφαρμογή ως ένα ενιαίο σύστημα και απεικονίζει τις αλληλεπιδράσεις της με τους βασικούς εξωτερικούς δρώντες και τα εξωτερικά συστήματα που βρίσκονται εκτός των ορίων της (system boundary).

Στο κέντρο του διαγράμματος βρίσκεται η πλατφόρμα ThesisApp, η οποία αναπαρίσταται ως ενιαίο σύστημα χωρίς εσωτερική αποδόμηση. Τα επιμέρους υποσυστήματα βάση δεδομένων, αποθετήριο αντικειμένων, υπηρεσία email, πλατφόρμα ενορχήστρωσης αποτελούν εσωτερικά δομικά στοιχεία που ενορχηστρώνονται ως μέρος της συνολικής ανάπτυξης και αναλύονται στο διάγραμμα συστατικών (Container Diagram) (δηλ. το διάγραμμα της αρχιτεκτονικής της πλατφόρμας μας). Οι κύριοι ανθρώπινοι δρώντες του συστήματος είναι ο Χρήστης (User) και ο Διαχειριστής (Admin). Ο χρήστης αλληλεπιδρά με την εφαρμογή μέσω web περιβάλλοντος και έχει τη δυνατότητα να διαχειρίζεται σύνολα δεδομένων, να αιτείται εκπαιδεύσεις, να εκτελεί προβλέψεις και να διαχειρίζεται τους παραγόμενους πόρους (μοντέλα, εκτελέσεις, αλγορίθμους). Ο διαχειριστής διαθέτει όλα τα δικαιώματα του απλού χρήστη και επιπλέον αναλαμβάνει λειτουργίες εποπτείας, όπως η διαχείριση χρηστών και η έγκριση κατηγοριών.

Το μοναδικό εξωτερικό σύστημα με το οποίο αλληλεπιδρά η εφαρμογή είναι το Container Registry (π.χ. DockerHub), από το οποίο ανακτώνται οι εικόνες δοχείων των προσαρμοσμένων αλγορίθμων που έχουν δημοσιευθεί από τους χρήστες. Όλα τα υπόλοιπα στοιχεία — συμπεριλαμβανομένου του Kubernetes Cluster, της βάσης δεδομένων, του αποθετηρίου αντικειμένων και της υπηρεσίας email — αναπτύσσονται και ενορχηστρώνονται ως αναπόσπαστα μέρη της λύσης και βρίσκονται εντός των ορίων του συστήματος.

Το διάγραμμα καταδεικνύει ότι η εφαρμογή λειτουργεί ως κεντρικός κόμβος συντονισμού, ο οποίος ενοποιεί διεπαφές χρήστη, υπολογιστικούς πόρους και μηχανισμούς αποθήκευσης, διατηρώντας σαφή όρια μεταξύ του συστήματος και του εξωτερικού περιβάλλοντος.



Εικόνα 4.1: Διάγραμμα Περιβάλλοντος

4.2.2 Αρχιτεκτονική Συστήματος και Διάγραμμα Συστατικών

Η εφαρμογή υιοθετεί μια πολυεπίπεδη λογική αρχιτεκτονική (layered architecture), η οποία οργανώνεται σε τέσσερα κύρια επίπεδα: παρουσίασης (presentation), ελεγκτών (controllers), επιχειρησιακής λογικής (services) και πρόσβασης δεδομένων (repositories). Η αρχιτεκτονική αυτή (δείτε Εικόνα 4.2) εξασφαλίζει σαφή διαχωρισμό ευθυνών μεταξύ των επιπέδων και ελαχιστοποιεί τις αλληλεξαρτήσεις, διευκολύνοντας τη συντηρησιμότητα και την επεκτασιμότητα του συστήματος.

- **Επίπεδο Παρουσίασης (Frontend)**

Το ανώτερο επίπεδο της αρχιτεκτονικής αποτελείται από την εφαρμογή frontend, η οποία υλοποιεί τη διεπαφή χρήστη ως single-page application. Τα βασικά συστατικά του frontend οργανώνονται σε λειτουργικές ενότητες (feature modules), καθεμία από τις οποίες αντιστοιχεί σε μια βασική λειτουργία του συστήματος:

- *Training Module*: Φόρμες εκπαίδευσης για προκαθορισμένους αλγορίθμους, προσαρμοσμένους αλγορίθμους και επανεκπαίδευση, με ενσωματωμένα συστατικά διαμόρφωσης παραμέτρων αλγορίθμων και επιλογής στηλών δεδομένων.
- *Model Module*: Περιήγηση, αναζήτηση, οριστικοποίηση και διαχείριση εκπαιδευμένων μοντέλων.
- *Dataset Module*: Μεταφόρτωση, διαχείριση και διαμοιρασμός συνόλων δεδομένων.
- *Execution Module*: Εκτέλεση προβλέψεων και λήψη αποτελεσμάτων.
- *Algorithm Module*: Δημιουργία και διαχείριση προσαρμοσμένων αλγορίθμων.
- *Category Module*: Υποβολή αιτημάτων κατηγοριών και πλοήγηση στην ιεραρχία κατηγοριών.
- *User & Admin Module*: Διαχείριση προφίλ, χρηστών και εγκρίσεων κατηγοριών.
- *Task Store*: Κεντρικός μηχανισμός παρακολούθησης ενεργών εργασιών, ο οποίος διατηρεί την κατάσταση μέσω polling και ενημερώνει τα συστατικά σε πραγματικό χρόνο.

Το frontend επικοινωνεί αποκλειστικά με το backend μέσω RESTful API κλήσεων.

- **Επίπεδο Ελεγκτών (Controllers)**

Οι ελεγκτές αποτελούν το σημείο εισόδου των αιτημάτων στο backend και είναι υπεύθυνοι για την επικύρωση εισόδου, τη δρομολόγηση προς τις κατάλληλες υπηρεσίες και τη μορφοποίηση των απαντήσεων. Τα βασικά συστατικά αυτού του επιπέδου είναι:

- *TrainingController*: Διαχείριση αιτημάτων εκπαίδευσης μοντέλων (προκαθορισμένων και προσαρμοσμένων).
- *ModelController*: Διαχείριση μοντέλων (αναζήτηση, οριστικοποίηση, επεξεργασία, διαγραφή).
- *ModelExecutionController*: Διαχείριση εκτελέσεων πρόβλεψης.
- *DatasetController*: Μεταφόρτωση και διαχείριση συνόλων δεδομένων.
- *AlgorithmController*: Δημιουργία και διαχείριση προσαρμοσμένων αλγορίθμων.
- *CategoryController*: Διαχείριση κατηγοριών και αιτημάτων έγκρισης.
- *UserController*: Διαχείριση χρηστών και ρόλων (λειτουργίες διαχειριστή).
- *AsyncTaskStatusController*: Παρακολούθηση κατάστασης και ακύρωση ασύγχρονων εργασιών.

- **Επίπεδο Υπηρεσιών (Services)**

Το επίπεδο υπηρεσιών υλοποιεί τον πυρήνα της επιχειρησιακής λογικής και τον συντονισμό των ροών εργασίας. Τα βασικά συστατικά περιλαμβάνουν:

- *StandardTrainingService*: Ενορχήστρωση εκπαίδευσης με προκαθορισμένους αλγορίθμους — προετοιμασία δεδομένων, δημιουργία εργασίας εκτέλεσης και συλλογή αποτελεσμάτων.
- *CustomTrainingService*: Ενορχήστρωση εκπαίδευσης με προσαρμοσμένους αλγορίθμους — φόρτωση Docker image, μεταβίβαση υπερπαραμέτρων και διαχείριση εκτέλεσης.
- *ModelExecutionService / CustomModelExecutionService*: Εκτέλεση προβλέψεων με εκπαιδευμένα μοντέλα (προκαθορισμένα και προσαρμοσμένα αντίστοιχα).
- *ModelService*: Διαχείριση κύκλου ζωής μοντέλων (αποθήκευση, οριστικοποίηση, αναζήτηση, διαγραφή).
- *DatasetService*: Διαχείριση συνόλων δεδομένων (μεταφόρτωση, ανάκτηση, διαγραφή).
- *AlgorithmService / CustomAlgorithmService*: Διαχείριση προκαθορισμένων και προσαρμοσμένων αλγορίθμων.
- *CategoryService*: Διαχείριση κατηγοριών, ιεραρχίας και ροής έγκρισης.
- *UserService / AuthService*: Αυθεντικοποίηση, εξουσιοδότηση, διαχείριση χρηστών και JWT tokens.
- *TaskStatusService*: Παρακολούθηση κατάστασης ασύγχρονων εργασιών και ακύρωση.
- *ObjectStorageService*: Αφαίρεση αλληλεπίδρασης με το αποθετήριο αντικειμένων (αποθήκευση/ανάκτηση αρχείων δεδομένων, μοντέλων και αποτελεσμάτων).
- *ContainerJobRunner*: Αφαίρεση δημιουργίας και διαχείρισης εργασιών στην πλατφόρμα ενορχήστρωσης.
- *EmailService*: Αποστολή ηλεκτρονικού ταχυδρομείου για λειτουργίες όπως η επαναφορά κωδικού.

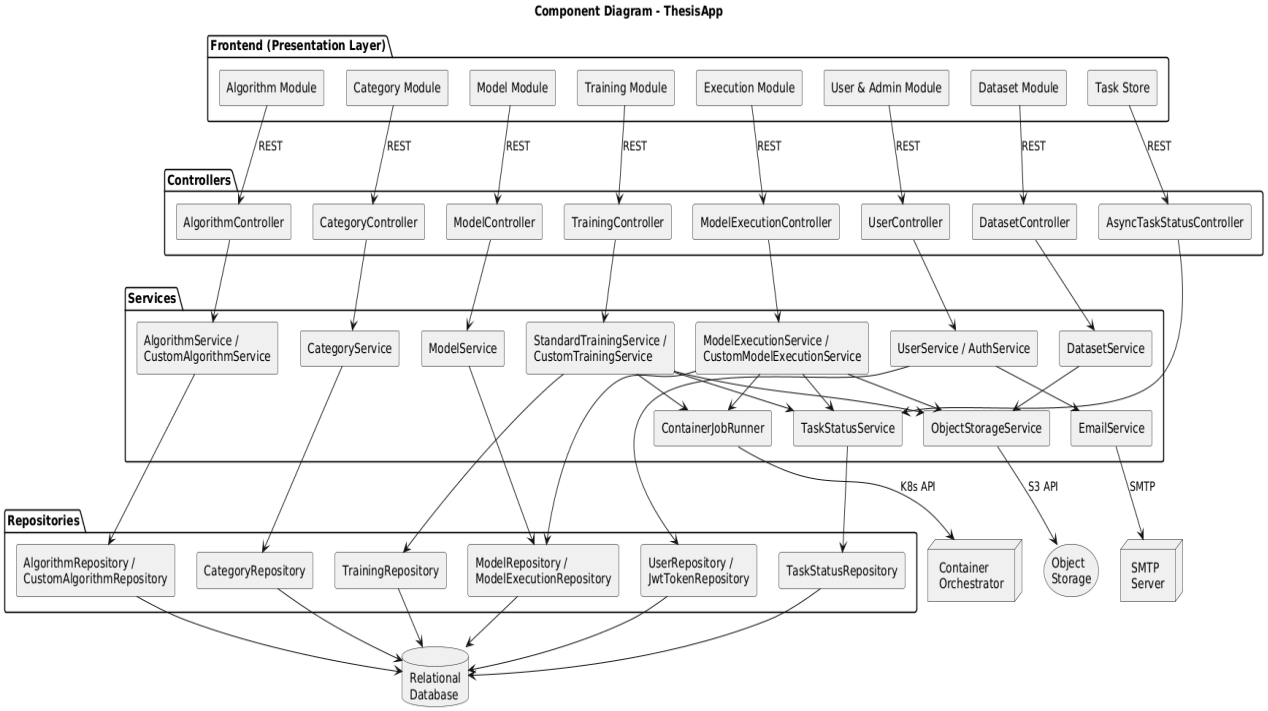
- **Επίπεδο Αποθετηρίων (Repositories)**

Το επίπεδο αποθετηρίων παρέχει αφαιρετική πρόσβαση στη βάση δεδομένων, απομονώνοντας τα υπόλοιπα επίπεδα από τις λεπτομέρειες αποθήκευσης. Τα κυριότερα αποθετήρια είναι:

- *UserRepository / RoleRepository*: Διαχείριση χρηστών και ρόλων.

- *TrainingRepository*: Αποθήκευση εκπαιδεύσεων και αποτελεσμάτων.
- *ModelRepository / ModelExecutionRepository*: Διαχείριση μοντέλων και εκτελέσεων πρόβλεψης.
- *AlgorithmRepository / CustomAlgorithmRepository*: Διαχείριση αλγορίθμων και διαμορφώσεών τους.
- *CategoryRepository / CategoryRequestRepository*: Διαχείριση κατηγοριών και αιτημάτων έγκρισης.
- *TaskStatusRepository*: Παρακολούθηση κατάστασης εργασιών.
- *JwtTokenRepository*: Διαχείριση JWT tokens και ανακλήσεων.

Τα αποθετήρια επικοινωνούν με τη σχεσιακή βάση δεδομένων για τα μεταδεδομένα, ενώ η ObjectStorageService επικοινωνεί με το αποθετήριο αντικειμένων (Object Storage) για τα αρχεία. Η EmailService επικοινωνεί με τον SMTP server για την αποστολή email. Ο ContainerJobRunner επικοινωνεί με την πλατφόρμα ενορχήστρωσης (Container Orchestrator) μέσω του Kubernetes API για τη δημιουργία, παρακολούθηση και διαχείριση εργασιών εκπαίδευσης και πρόβλεψης σε απομονωμένα δοχεία.



Εικόνα 4.2: Διάγραμμα Συστατικών

4.2.3 Διαγράμματα Περιπτώσεων Χρήσης

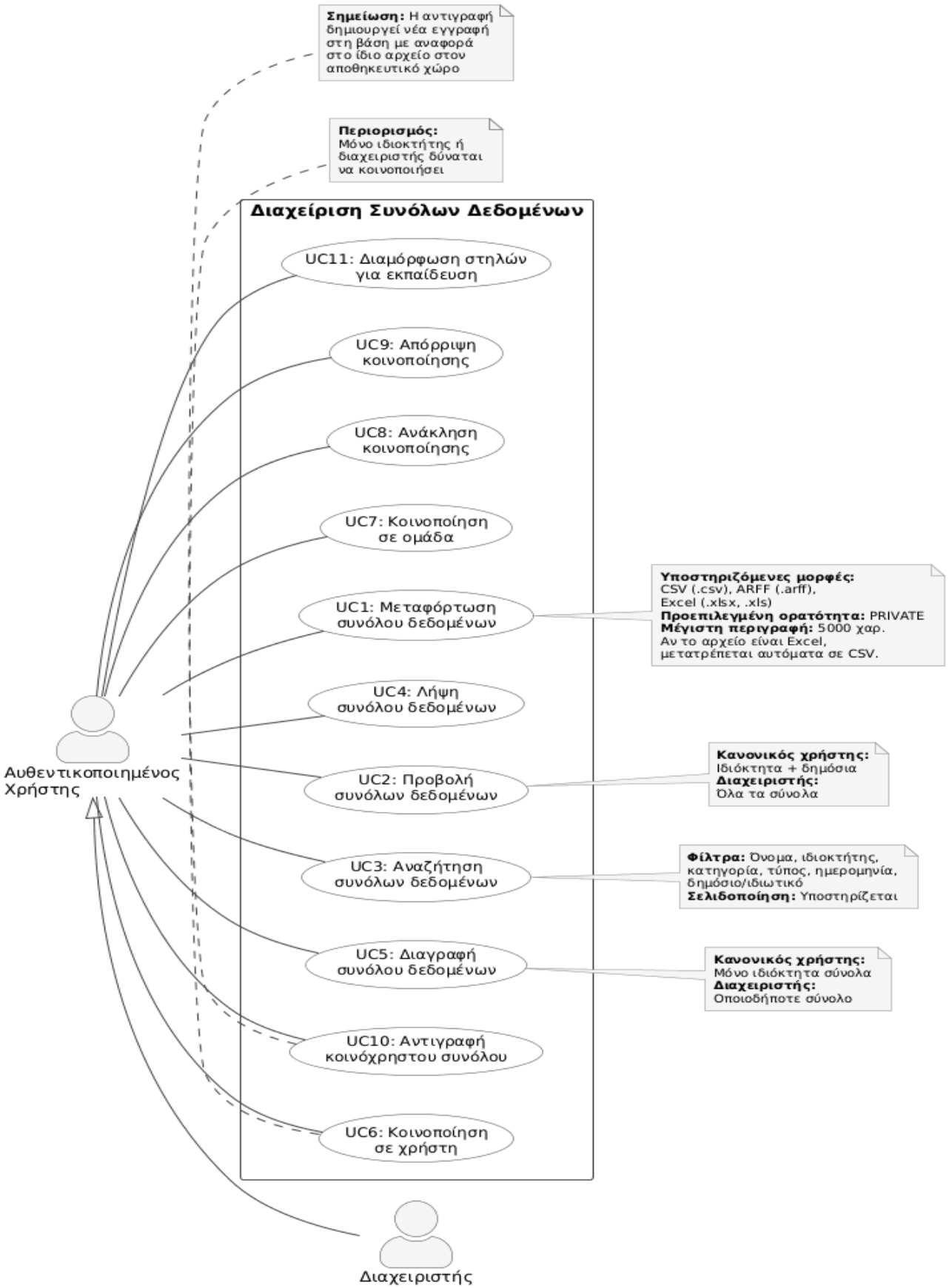
Τα διαγράμματα περιπτώσεων χρήσης αποτυπώνουν τις λειτουργικές δυνατότητες του συστήματος από τη σκοπιά των χρηστών και περιγράφουν τις βασικές αλληλεπιδράσεις μεταξύ των δρώντων (actors) / εξωτερικών συστημάτων και της εφαρμογής. Στο συνολικό διάγραμμα περιπτώσεων χρήσης διακρίνονται σαφώς οι λειτουργίες που είναι διαθέσιμες σε όλους τους χρήστες (ανεξάρτητα από τους ρόλους τους) και εκείνες που προορίζονται αποκλειστικά για διαχειριστές.

Οι βασικές κατηγορίες λειτουργιών περιλαμβάνουν την αυθεντικοποίηση χρηστών, τη διαχείριση συνόλων δεδομένων, την εκπαίδευση και εκτέλεση μοντέλων, τη διαχείριση μοντέλων και αλγορίθμων, καθώς και τη διαχείριση κατηγοριών. Οι διαχειριστές διαθέτουν επιπλέον λειτουργίες που αφορούν τη διαχείριση χρηστών και την έγκριση ή απόρριψη αιτημάτων κατηγοριοποίησης.

Η αναλυτική περιγραφή επιλεγμένων περιπτώσεων χρήσης, όπως η εκπαίδευση μοντέλου με Weka, η εκτέλεση πρόβλεψης και η δημιουργία προσαρμοσμένου αλγορίθμου, επιτρέπει την κατανόηση τόσο της κύριας ροής όσο και πιθανών εναλλακτικών σεναρίων, όπως ακύρωση εκτέλεσης ή μη συμβατότητα μοντέλου. Με τον τρόπο αυτό, τα use cases λειτουργούν ως γέφυρα μεταξύ απαιτήσεων και υλοποίησης, προσφέροντας σαφή καθοδήγηση για τον σχεδιασμό των επιμέρους υποσυστημάτων.

4.2.3.1 Διαχείριση Συνόλου Δεδομένων

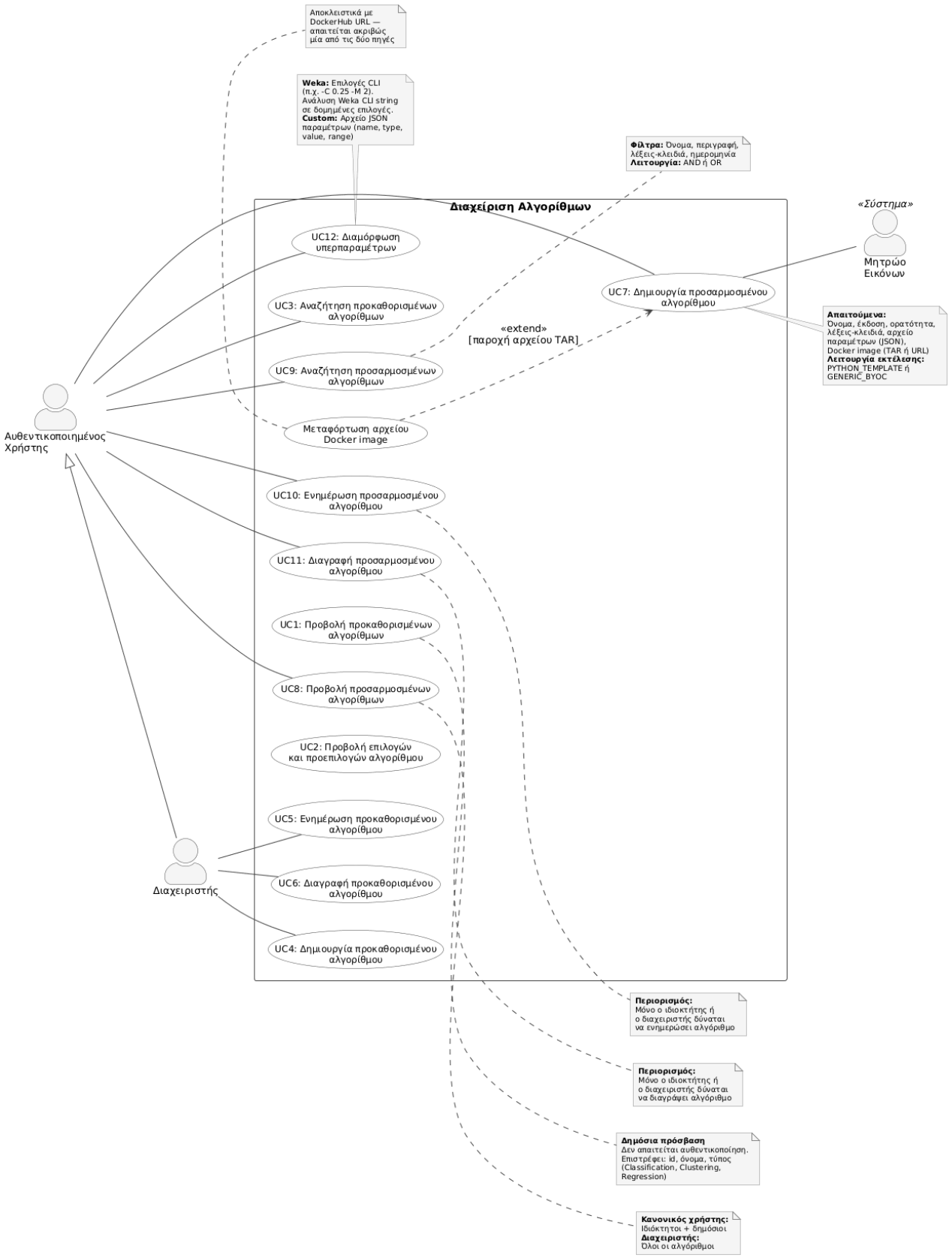
Το διάγραμμα περιπτώσεων χρήσης Διαχείριση Συνόλου Δεδομένων (δείτε Εικόνα 4.3) αποτυπώνει τις λειτουργίες μεταφόρτωσης, προβολής, αναζήτησης, λήψης, διαγραφής και κοινοποίησης συνόλων δεδομένων. Οι βασικοί δρώντες είναι ο αυθεντικοποιημένος χρήστης και ο διαχειριστής. Η μεταφόρτωση περιλαμβάνει ως εσωτερικές λειτουργίες την επικύρωση αρχείου (υποστηριζόμενες μορφές: CSV, ARFF, Excel), τη δημιουργία μοναδικού ονόματος αρχείου και την αποθήκευση στο MinIO, ενώ η μετατροπή μορφής Excel σε CSV αποτελεί επέκταση (extend) που ενεργοποιείται μόνο για αρχεία .xlsx ή .xls. Ο χρήστης δύναται επίσης να κοινοποιήσει σύνολα δεδομένων σε μεμονωμένους χρήστες ή ομάδες, να ανακαλέσει ή να απορρίψει κοινοποιήσεις, να αντιγράψει κοινόχρηστα σύνολα και να διαμορφώσει στήλες για εκπαίδευση. Η πρόσβαση ελέγχεται μέσω δικαιωμάτων ιδιοκτησίας, ενώ ο διαχειριστής έχει δυνατότητα προβολής όλων των συνόλων και διαγραφής οποιουδήποτε. Κάθε ενέργεια κοινοποίησης καταγράφεται σε ιστορικό.



Εικόνα 4.3 Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Συνόλου Δεδομένων

4.2.3.2 Διαχείριση Αλγορίθμων

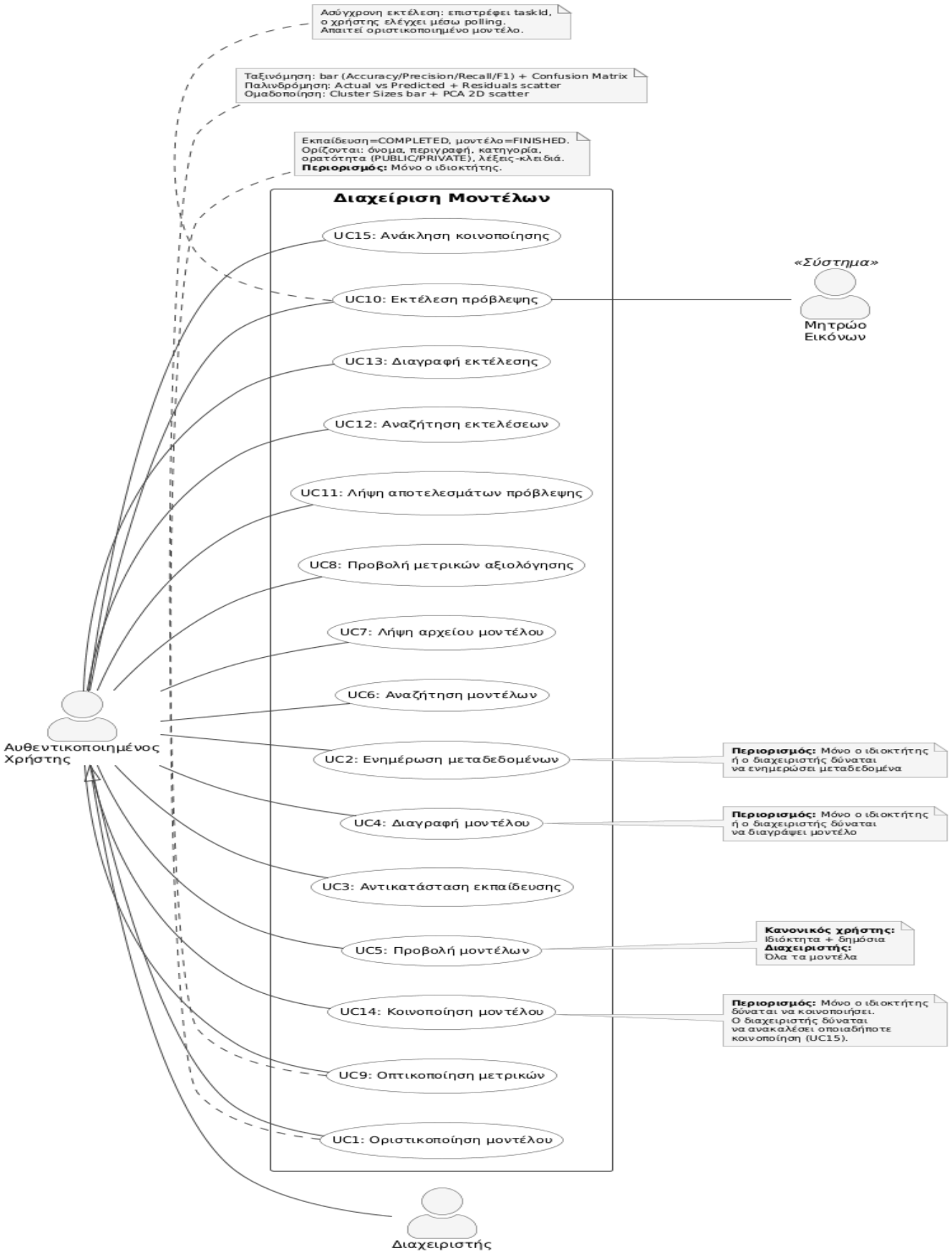
Το διάγραμμα περιπτώσεων χρήσης Διαχείρισης Αλγορίθμων (δείτε Εικόνα 4.4) περιγράφει τη διαχείριση δύο κατηγοριών αλγορίθμων: των προκαθορισμένων αλγορίθμων Weka και των προσαρμοσμένων αλγορίθμων (custom). Οι προκαθορισμένοι αλγόριθμοι είναι δημοσίως προσβάσιμοι ακόμα και χωρίς αυθεντικοποίηση, υποστηρίζοντας προβολή λίστας, προβολή επιλογών με τιμές προεπιλογής, και αναζήτηση κατά όνομα ή περιγραφή. Η δημιουργία, ενημέρωση και διαγραφή προκαθορισμένων αλγορίθμων αποτελεί αποκλειστικό προνόμιο του διαχειριστή. Οι προσαρμοσμένοι αλγόριθμοι δημιουργούνται από τον αυθεντικοποιημένο χρήστη μέσω μεταφόρτωσης Docker image (TAR αρχείο στο MinIO ή αναφορά URL DockerHub) και αρχείου παραμέτρων JSON, υποστηρίζοντας δύο λειτουργίες εκτέλεσης: PYTHON_TEMPLATE (έκχυση σεναρίων εκπαίδευσης/πρόβλεψης) και GENERIC_BYOC (εκτέλεση του ENTRYPOINT του container χωρίς παρέμβαση). Ο ιδιοκτήτης ενός προσαρμοσμένου αλγορίθμου δύναται να τον ενημερώσει ή να τον διαγράψει. Ως εσωτερικές λειτουργίες αποτυπώνονται η επικύρωση δεδομένων, η ανάλυση αρχείου παραμέτρων, η ανάλυση επιλογών Weka CLI, και ο έλεγχος δικαιωμάτων πρόσβασης.



Εικόνα 4.4: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Αλγορίθμων

4.2.3.3 Διαχείριση Μοντέλων

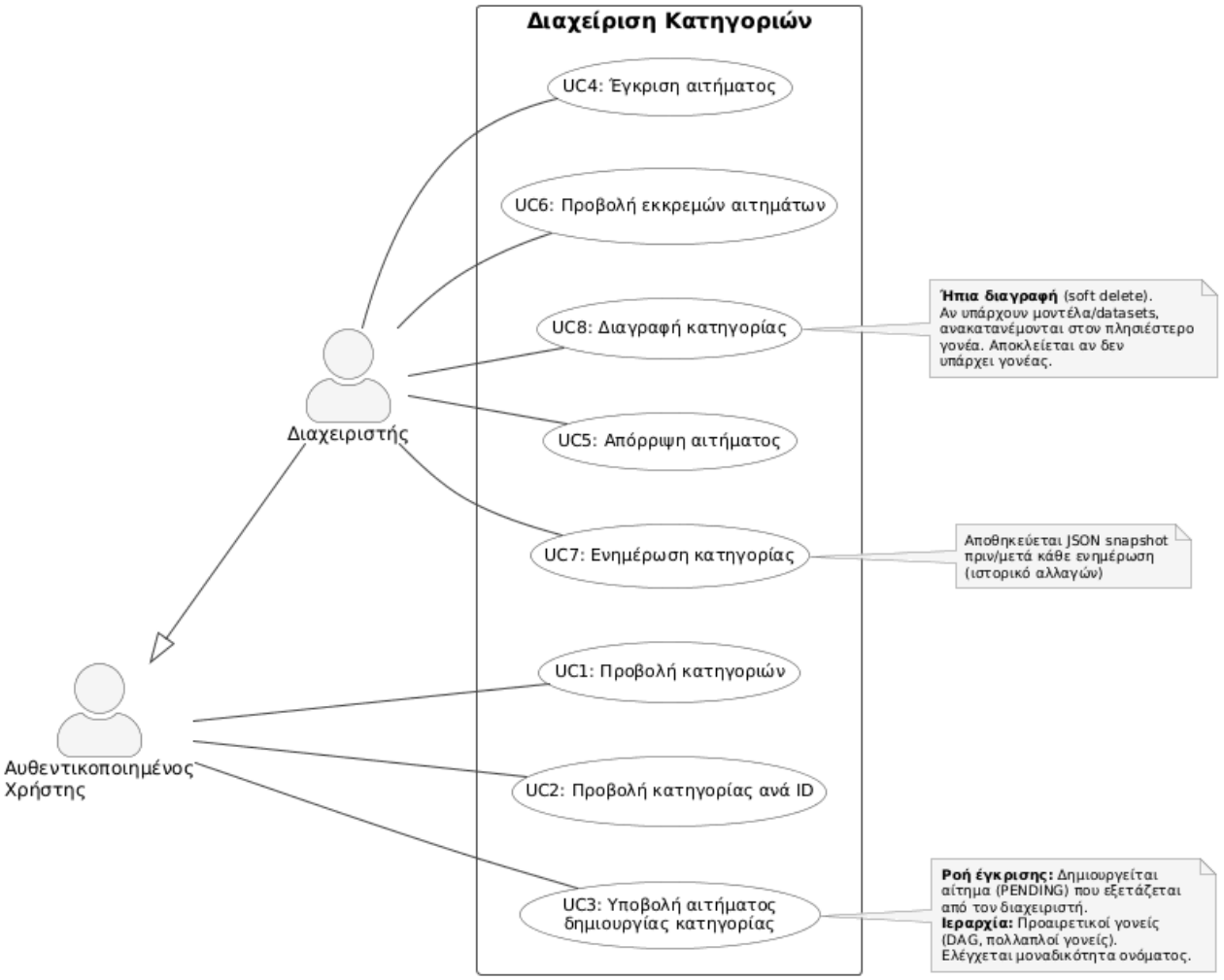
Το διάγραμμα περιπτώσεων χρήσης Διαχείρισης Μοντέλων (δείτε Εικόνα 4.5) απεικονίζει τον πλήρη κύκλο ζωής των μοντέλων μηχανικής μάθησης. Μετά την ολοκλήρωση μιας εκπαίδευσης, το μοντέλο δημιουργείται αυτόματα σε κατάσταση μη οριστικοποιημένου και ο ιδιοκτήτης δύναται να το οριστικοποιήσει ορίζοντας μεταδεδομένα (όνομα, περιγραφή, κατηγορία, ορατότητα, λέξεις-κλειδιά). Μετά την οριστικοποίηση, υποστηρίζονται η ενημέρωση μεταδεδομένων, η αντικατάσταση της υποκείμενης εκπαίδευσης (διατηρώντας το ίδιο αναγνωριστικό), η λήψη του αρχείου μοντέλου και η διαγραφή. Ο αυθεντικοποιημένος χρήστης δύναται να προβάλλει τα προσβάσιμα μοντέλα (ιδιόκτητα, δημόσια και κοινοποιημένα), να τα αναζητήσει με φίλτρα (λειτουργία AND ή OR), να προβάλλει μετρικές αξιολόγησης και να εκτελέσει προβλέψεις ασύγχρονα. Η οπτικοποίηση μετρικών διαφοροποιείται ανάλογα με τον τύπο αλγορίθμου: διαγράμματα ακρίβειας και πίνακας σύγχυσης για ταξινόμηση, scatter plots για παλινδρόμηση, και διαγράμματα μεγεθών συστάδων με προβολή PCA για ομαδοποίηση. Ο ιδιοκτήτης δύναται επίσης να κοινοποιήσει μοντέλα σε χρήστες ή ομάδες και να ανακαλέσει κοινοποιήσεις, ενώ ο διαχειριστής έχει πρόσβαση σε όλα τα μοντέλα.



Εικόνα 4.5: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Μοντέλων

4.2.3.4 Διαχείριση Κατηγοριών

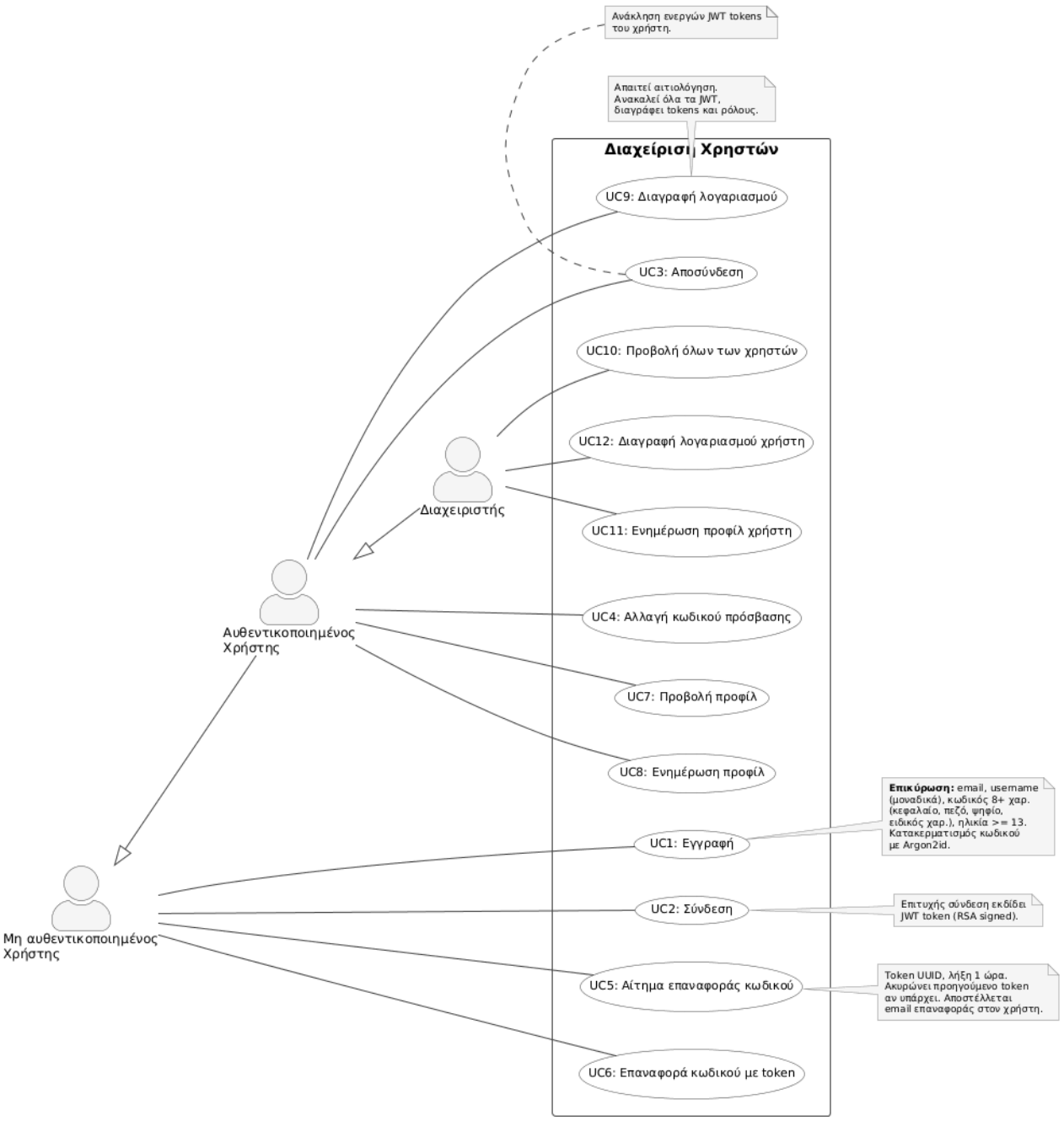
Το διάγραμμα περιπτώσεων χρήσης Διαχείρισης Κατηγοριών (δείτε Εικόνα 4.6) απεικονίζει τη διαχείριση του ιεραρχικού συστήματος κατηγοριοποίησης μοντέλων και συνόλων δεδομένων. Ο αυθεντικοποιημένος χρήστης δύναται να προβάλλει τις υφιστάμενες κατηγορίες και να υποβάλει αίτημα δημιουργίας νέας κατηγορίας, το οποίο τίθεται σε κατάσταση εκκρεμότητας (PENDING). Ο διαχειριστής εξετάζει τα εκκρεμή αιτήματα και δύναται να τα εγκρίνει ή να τα απορρίψει, να ενημερώσει υφιστάμενες κατηγορίες (με καταγραφή ιστορικού αλλαγών σε μορφή JSON snapshot πριν και μετά κάθε τροποποίηση) και να διαγράψει κατηγορίες μέσω ήπιας διαγραφής (soft delete).



Εικόνα 4.6: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Κατηγοριών

4.2.3.5 Διαχείριση Χρηστών

Το διάγραμμα περιπτώσεων χρήσης Διαχείρισης Χρηστών (δείτε Εικόνα 4.7) απεικονίζει τις βασικές λειτουργίες αυθεντικοποίησης και διαχείρισης λογαριασμών. Διακρίνονται τρεις δρώντες: ο μη αυθεντικοποιημένος χρήστης, ο αυθεντικοποιημένος χρήστης και ο διαχειριστής, με σχέση κληρονομικότητας μεταξύ τους. Ο μη αυθεντικοποιημένος χρήστης δύναται να πραγματοποιήσει εγγραφή, σύνδεση και επαναφορά κωδικού πρόσβασης μέσω αποστολής email με token περιορισμένης διάρκειας (1 ώρα). Ο αυθεντικοποιημένος χρήστης αποκτά πρόσβαση σε λειτουργίες προβολής και ενημέρωσης του προφίλ του, αλλαγής κωδικού πρόσβασης και διαγραφής λογαριασμού με υποχρεωτική αιτιολόγηση. Ο διαχειριστής διαθέτει πρόσθετες λειτουργίες προβολής όλων των χρηστών, ενημέρωσης και διαγραφής οποιουδήποτε λογαριασμού. Ως εξωτερικό σύστημα συμμετέχει η υπηρεσία ηλεκτρονικού ταχυδρομείου για την αποστολή συνδέσμων επαναφοράς κωδικού.



Εικόνα 4.7: Διάγραμμα Περιπτώσεων Χρήσης Διαχείρισης Χρηστών

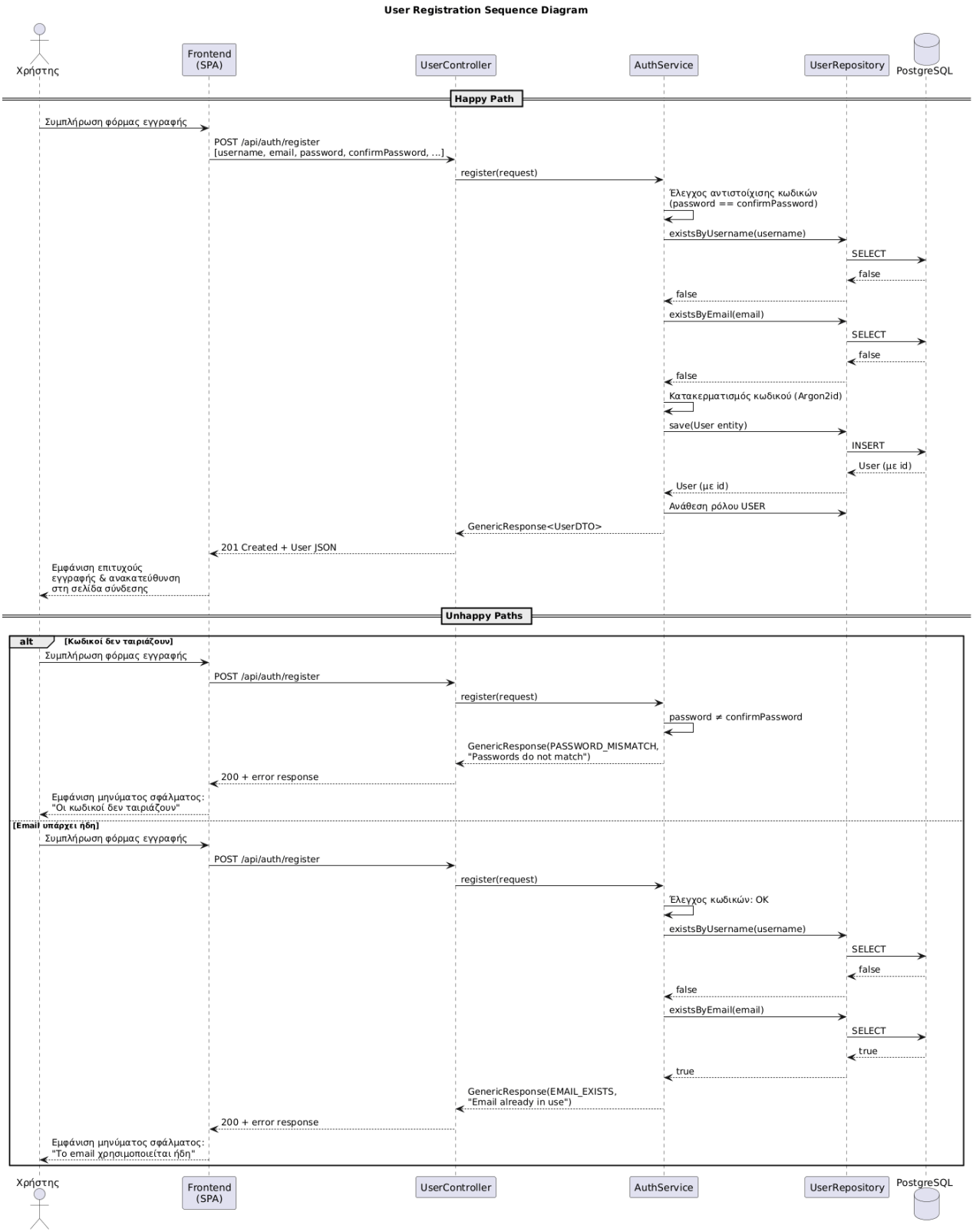
4.2.4 Διαγράμματα Ακολουθίας

Τα διαγράμματα ακολουθίας αποτυπώνουν τη χρονική σειρά των αλληλεπιδράσεων μεταξύ των βασικών συστατικών του συστήματος και των εξωτερικών δρώντων για τις κύριες λειτουργικές ροές. Στην παρούσα ενότητα παρουσιάζονται οκτώ διαγράμματα που καλύπτουν τις βασικές λειτουργίες του συστήματος: εγγραφή και αυθεντικοποίηση χρήστη, μεταφόρτωση συνόλου δεδομένων, διαμοιρασμό πόρων, δημιουργία και έγκριση

κατηγοριών, εκπαίδευση μοντέλων με προκαθορισμένους και προσαρμοσμένους αλγορίθμους, και εκτέλεση προβλέψεων.

4.2.4.1 Εγγραφή Χρήστη

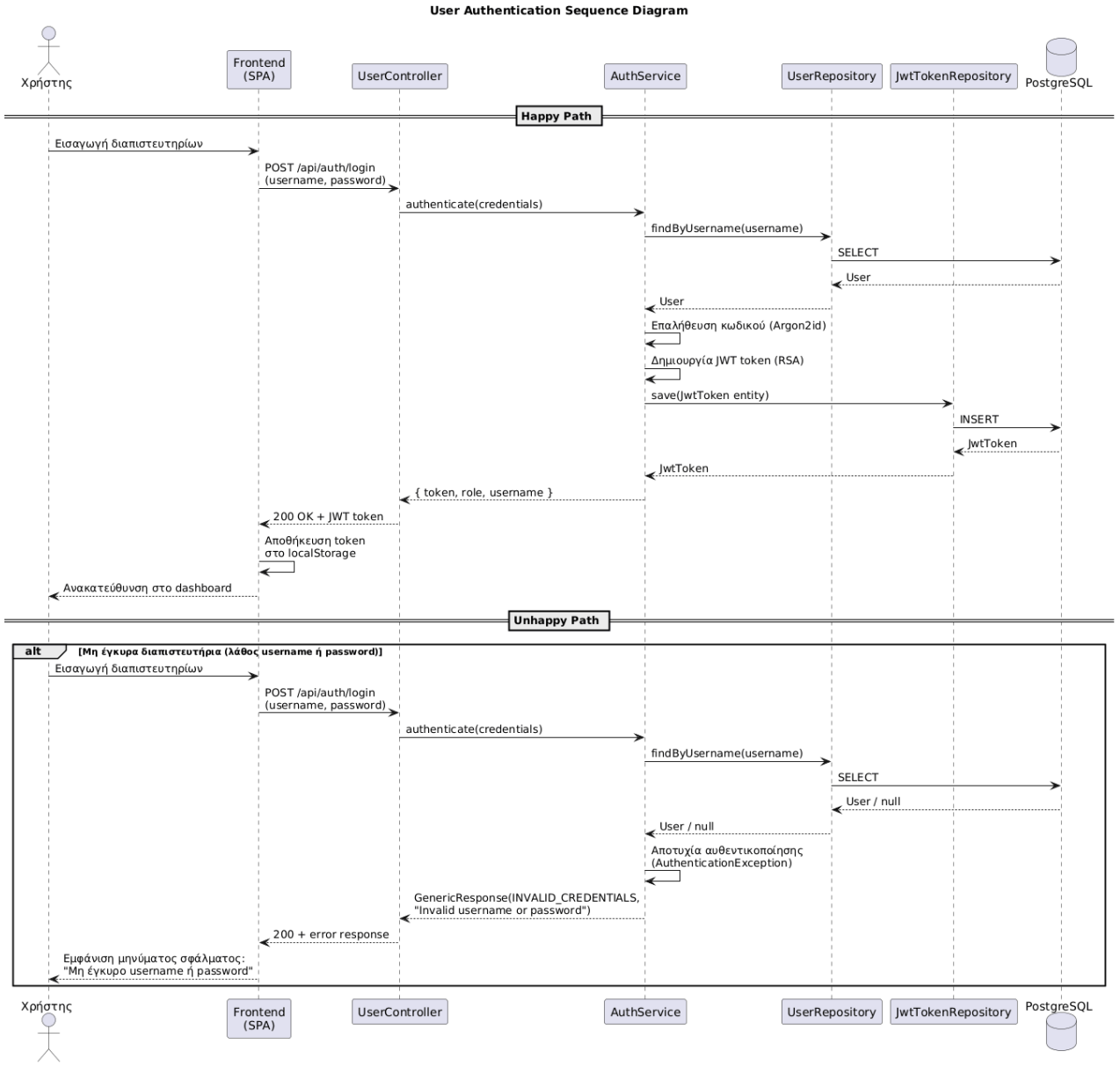
Το διάγραμμα ακολουθίας για την Εγγραφή Χρήστη (δείτε Εικόνα 4.8) απεικονίζει τη ροή εγγραφής νέου χρήστη στο σύστημα. Ο χρήστης μεταβαίνει στη σελίδα εγγραφής και υποβάλλει τα στοιχεία του μέσω της αντίστοιχης φόρμας. Το backend επικυρώνει τη μοναδικότητα του email και του ονόματος χρήστη, κατακερματίζει τον κωδικό πρόσβασης με τον αλγόριθμο Argon2id, αποθηκεύει τον νέο χρήστη στη βάση δεδομένων, αντιστοιχίζοντάς τον στον προεπιλεγμένο ρόλο (USER). Σε περίπτωση αποτυχίας της εγγραφής, το σύστημα διαχειρίζεται δύο πιθανά σενάρια σφάλματος. Πρώτον, εάν ο κωδικός πρόσβασης δεν ταιριάζει με τον επιβεβαιωτικό κωδικό, η υπηρεσία επιστρέφει απόκριση με κωδικό σφάλματος PASSWORD_MISMATCH χωρίς να πραγματοποιηθεί κάποια αναζήτηση στη βάση δεδομένων. Δεύτερον,, εάν η διεύθυνση email χρησιμοποιείται ήδη από άλλον χρήστη, επιστρέφεται κωδικός σφάλματος EMAIL_EXISTS. Σε όλες τις περιπτώσεις αποτυχίας, δεν δημιουργείται νέα εγγραφή χρήστη στη βάση δεδομένων και το frontend εμφανίζει κατάλληλο μήνυμα σφάλματος στον χρήστη.



Εικόνα 4.8: Διάγραμμα Ακολουθίας - Εγγραφή Χρήστη

4.2.4.2 Αυθεντικοποίηση Χρήστη

Το παρακάτω διάγραμμα ακολουθίας για την Αυθεντικοποίηση Χρήστη (δείτε Εικόνα 4.9) απεικονίζει τη ροή σύνδεσης χρήστη. Ο χρήστης μεταβαίνει στη σελίδα σύνδεσης, εισάγει τα διαπιστευτήριά του, το backend επικυρώνει τον κωδικό πρόσβασης, δημιουργεί ένα JWT token με ασύμμετρη κρυπτογράφηση RSA, το αποθηκεύει στη βάση δεδομένων και το επιστρέφει στο frontend, το οποίο το αποθηκεύει τοπικά για χρήση σε επόμενα αιτήματα. Σε περίπτωση αποτυχίας αυθεντικοποίησης, όπως η εισαγωγή λανθασμένου ονόματος χρήστη ή κωδικού πρόσβασης, η υπηρεσία AuthService εγείρει εξαίρεση αυθεντικοποίησης (AuthenticationException) και επιστρέφει απόκριση με κωδικό σφάλματος INVALID_CREDENTIALS. Το frontend εμφανίζει κατάλληλο μήνυμα σφάλματος στον χρήστη χωρίς να αποκαλύπτει ποιο από τα δύο πεδία είναι εσφαλμένο, αποτρέποντας πιθανές επιθέσεις απαρίθμησης χρηστών (user enumeration). Σε αυτή την περίπτωση, δεν δημιουργείται JWT token και δεν πραγματοποιείται καμία εγγραφή στη βάση δεδομένων.

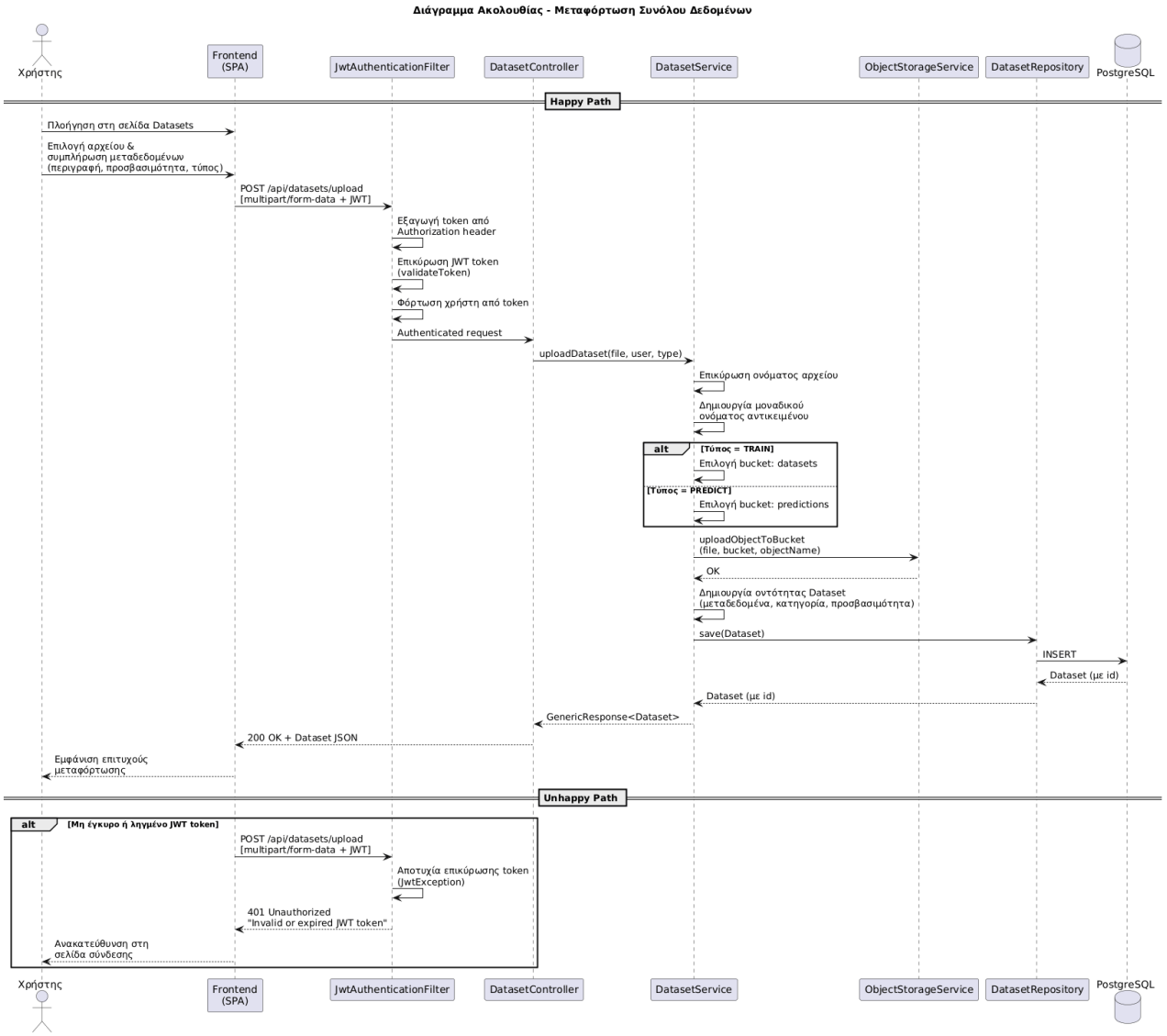


Εικόνα 4.9: Διάγραμμα Ακολουθίας - Αυθεντικοποίηση Χρήστη

4.2.4.3 Μεταφόρτωση Συνόλου Δεδομένων

Το διάγραμμα ακολουθίας Μεταφόρτωσης Συνόλου Δεδομένων (δείτε Εικόνα 4.10) απεικονίζει τη ροή μεταφόρτωσης ενός νέου συνόλου δεδομένων. Ο χρήστης μεταβαίνει στη σελίδα Datasets, επιλέγει αρχείο και συμπληρώνει τα μεταδεδομένα (περιγραφή, προσβασιμότητα, τύπος). Το backend επικυρώνει τη μορφή του αρχείου, αποθηκεύει το αρχείο στο αποθετήριο αντικειμένων, καταχωρίζει τα μεταδεδομένα στη βάση δεδομένων και επιστρέφει τα στοιχεία του νέου συνόλου δεδομένων. Πριν από την επεξεργασία του

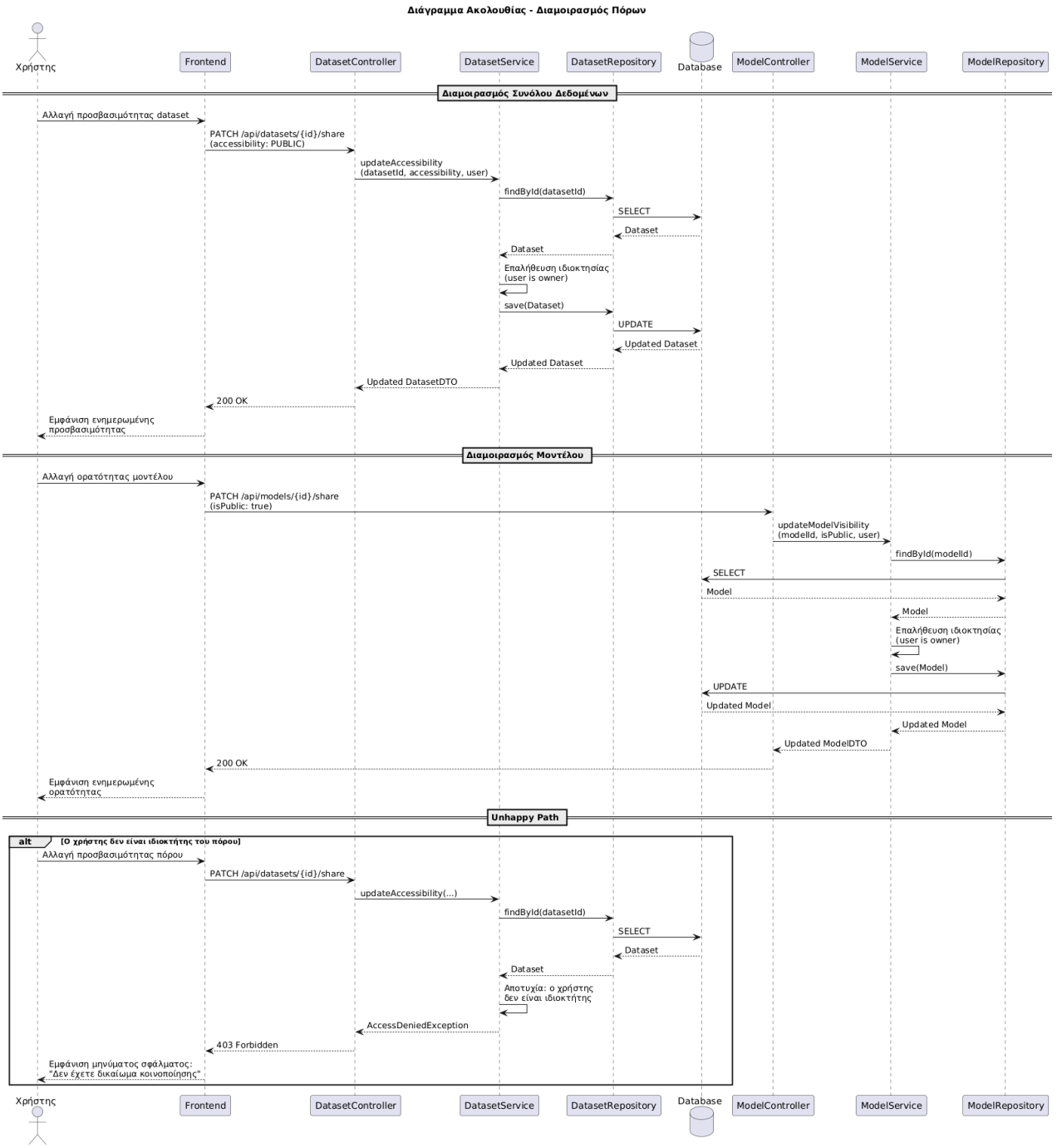
αιτήματος, το σύστημα εκτελεί επικύρωση του JWT token μέσω του JwtAuthenticationFilter. Το φίλτρο εξάγει το token από την κεφαλίδα Authorization, επαληθεύει την εγκυρότητά του μέσω του JwtTokenProvider και φορτώνει τα στοιχεία του χρήστη. Σε περίπτωση μη έγκυρου ή ληγμένου token, το σύστημα επιστρέφει απόκριση 401 Unauthorized χωρίς να προχωρήσει στην επεξεργασία του αιτήματος, και το frontend ανακατευθύνει τον χρήστη στη σελίδα σύνδεσης. Ο μηχανισμός αυτός εφαρμόζεται σε όλα τα προστατευμένα endpoints του συστήματος.



Εικόνα 4.10: Διάγραμμα Ακολουθίας - Μεταφόρτωση Συνόλου Δεδομένων

4.2.4.4 Διαμοιρασμός Πόρων

Το διάγραμμα ακολουθίας Διαμοιρασμού Πόρων (δείτε Εικόνα 4.11) απεικονίζει τη ροή αλλαγής προσβασιμότητας ενός πόρου (σύνολο δεδομένων ή μοντέλο). Ο ιδιοκτήτης τροποποιεί το επίπεδο προσβασιμότητας (Private, Public, Shared) μέσω της αντίστοιχης σελίδας διαχείρισης. Το backend επικυρώνει ότι ο χρήστης είναι ο ιδιοκτήτης του πόρου και ενημερώνει τη βάση δεδομένων, καθιστώντας τον πόρο διαθέσιμο ή μη σε άλλους χρήστες. Σε περίπτωση που ο χρήστης δεν είναι ο ιδιοκτήτης του πόρου, η υπηρεσία απορρίπτει το αίτημα και επιστρέφει απόκριση 403 Forbidden, εμποδίζοντας μη εξουσιοδοτημένες τροποποιήσεις στην προσβασιμότητα πόρων άλλων χρηστών.



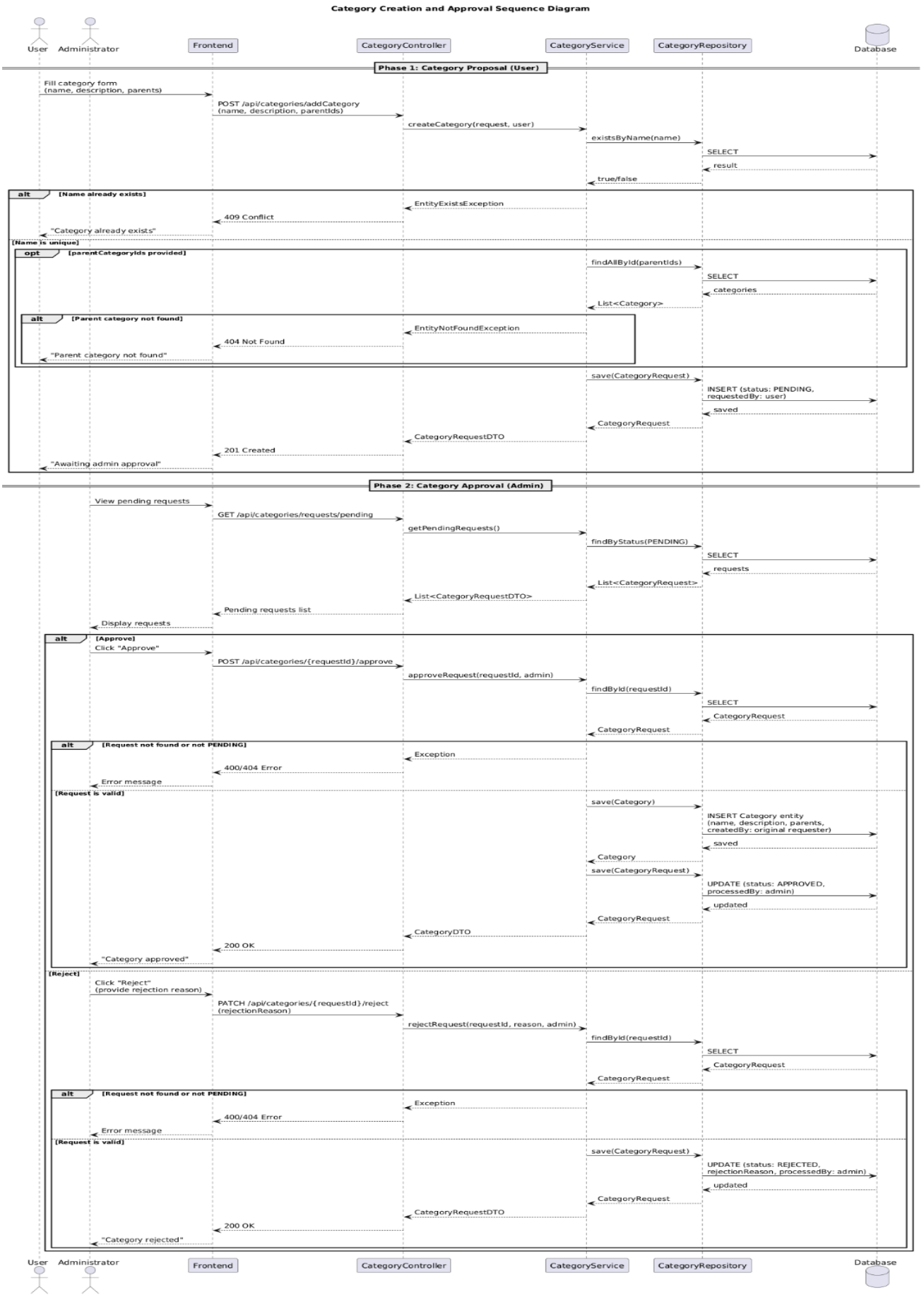
Εικόνα 4.11: Διάγραμμα Ακολουθίας - Διαμοιρασμός Πόρων

4.2.4.5 Διαχείριση Κατηγοριών

Το διάγραμμα ακολουθίας Διαχείρισης Κατηγοριών (δείτε Εικόνα 4.12) απεικονίζει την πλήρη ροή δημιουργίας κατηγορίας, η οποία περιλαμβάνει δύο φάσεις. Κατά την πρώτη φάση, ο χρήστης μεταβαίνει στη σελίδα Categories, υποβάλλει αίτημα δημιουργίας

κατηγορίας (όνομα, περιγραφή, γονικές κατηγορίες) και δημιουργείται αίτημα σε κατάσταση PENDING. Σε περίπτωση που υπάρχει ήδη κατηγορία με το ίδιο όνομα, η υπηρεσία απορρίπτει το αίτημα και επιστρέφει απόκριση 409 Conflict. Ομοίως, εάν κάποια από τις γονικές κατηγορίες δεν βρεθεί, επιστρέφεται απόκριση 404 Not Found.

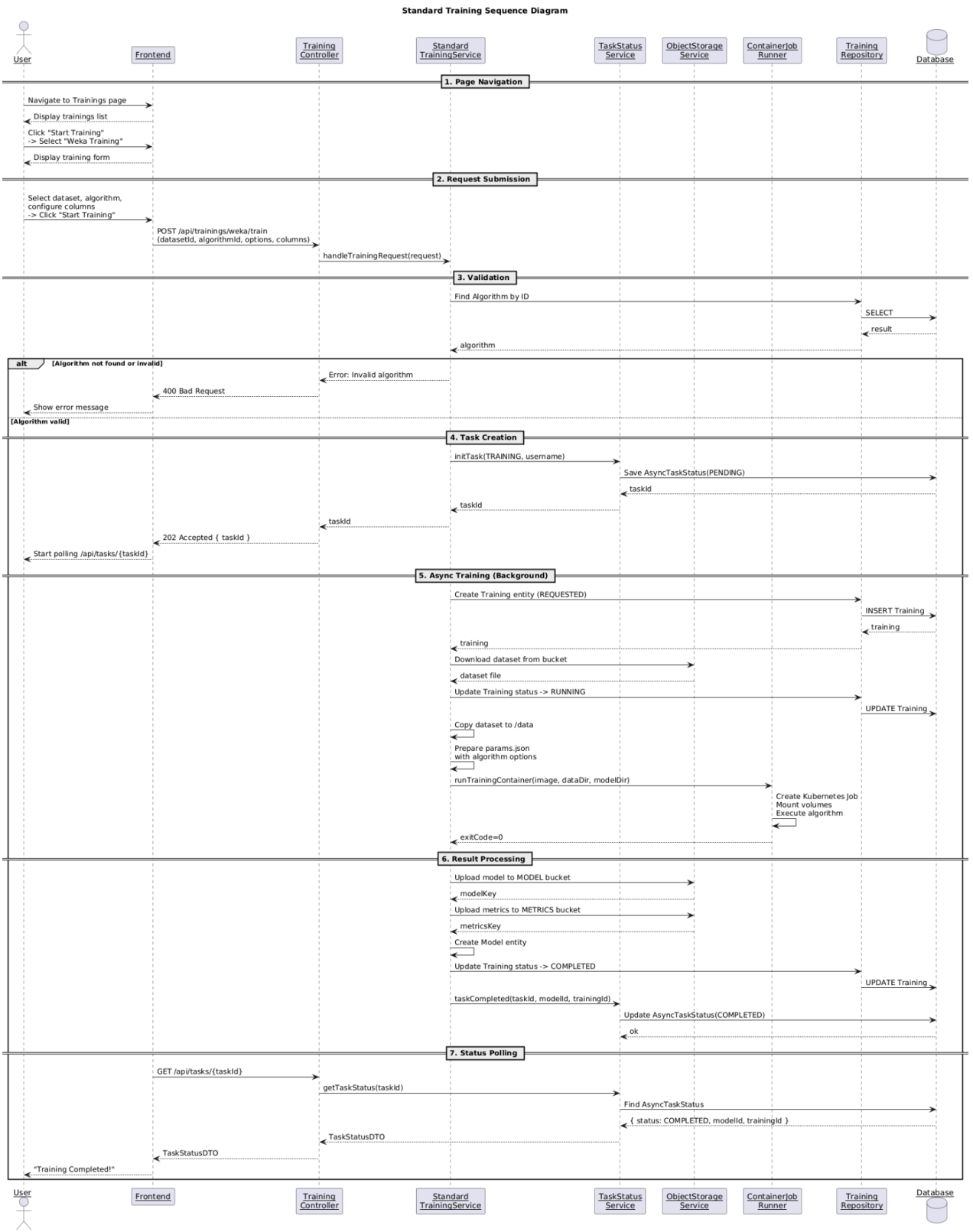
Κατά τη δεύτερη φάση, ο διαχειριστής μεταβαίνει στη σελίδα Approvals και εξετάζει τα εκκρεμή αιτήματα. Εάν εγκρίνει το αίτημα, δημιουργείται η κατηγορία στη βάση δεδομένων και το αίτημα ενημερώνεται σε κατάσταση APPROVED. Εάν ο διαχειριστής απορρίψει το αίτημα, καταχωρείται ο λόγος απόρριψης και το αίτημα ενημερώνεται σε κατάσταση REJECTED, χωρίς να δημιουργηθεί νέα κατηγορία. Και στις δύο περιπτώσεις, εάν το αίτημα δεν βρεθεί ή δεν βρίσκεται σε κατάσταση PENDING, η υπηρεσία επιστρέφει κατάλληλο μήνυμα σφάλματος.



Εικόνα 4.12: Διάγραμμα Ακολουθίας - Διαχείριση Κατηγοριών

4.2.4.6 Εκπαίδευση με Προκαθορισμένο Αλγόριθμο

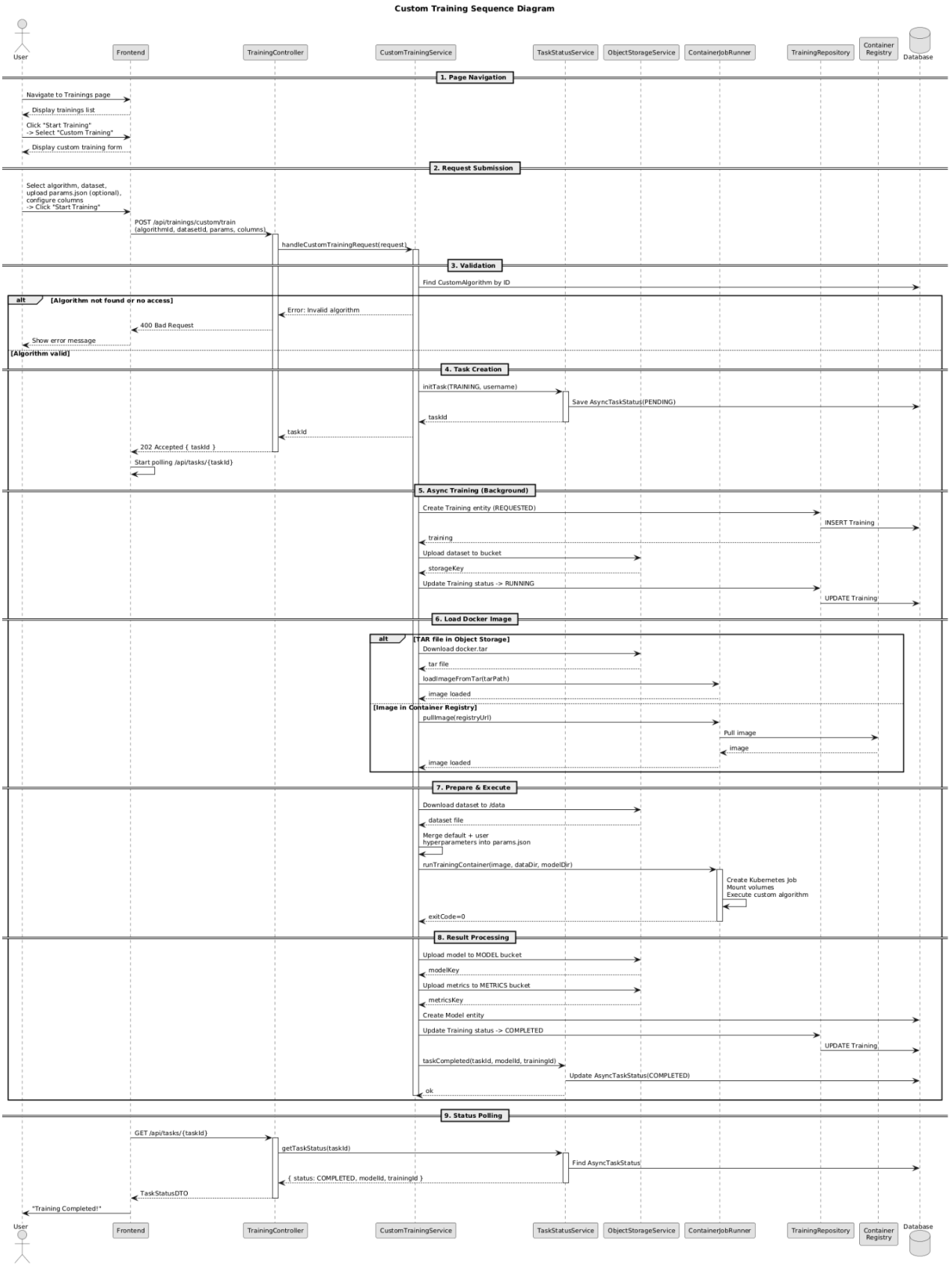
Το διάγραμμα ακολουθίας Εκπαίδευσης Προκαθορισμένου Αλγόριθμου (δείτε Εικόνα 4.13) απεικονίζει τη ροή εκπαίδευσης μοντέλου με προκαθορισμένο αλγόριθμο. Ο χρήστης μεταβαίνει στη σελίδα Trainings, επιλέγει τον τύπο εκπαίδευσης και συμπληρώνει τη φόρμα. Με την υποβολή του αιτήματος, το backend επικυρώνει τον αλγόριθμο και το σύνολο δεδομένων, δημιουργεί ασύγχρονη εργασία και επιστρέφει αναγνωριστικό εργασίας (taskId) στο frontend, το οποίο ξεκινά περιοδική ανίχνευση κατάστασης (polling). Στο παρασκήνιο, η υπηρεσία αποθηκεύει τα μεταδεδομένα της εκπαίδευσης στη βάση δεδομένων, ανακτά το σύνολο δεδομένων από το αποθετήριο αντικειμένων, προετοιμάζει τις παραμέτρους εκπαίδευσης και δημιουργεί ένα Kubernetes Job, το οποίο εκτελεί τον αλγόριθμο σε απομονωμένο container. Με την ολοκλήρωση, το παραγόμενο μοντέλο και οι μετρικές αξιολόγησης αποθηκεύονται στο αποθετήριο αντικειμένων, ενημερώνεται η βάση δεδομένων και η κατάσταση της εργασίας μεταβαίνει σε COMPLETED.



Εικόνα 4.13: Διάγραμμα Ακολουθίας - Εκπαίδευση Προκαθορισμένου Αλγόριθμου

4.2.4.7 Εκπαίδευση με Προσαρμοσμένο Αλγόριθμο

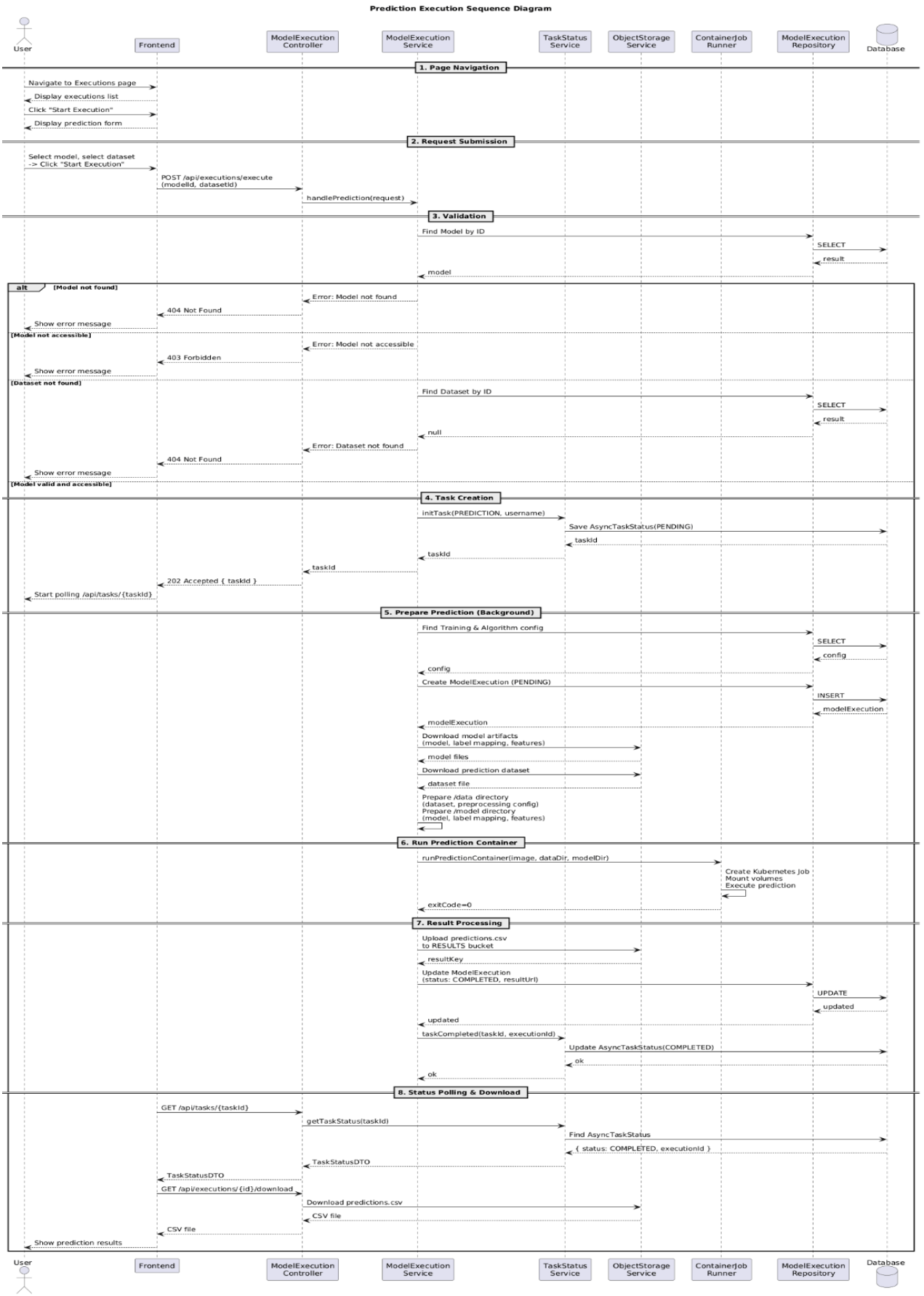
Το διάγραμμα ακολουθίας για την Εκπαίδευση Προσαρμοσμένου Αλγόριθμου (δείτε Εικόνα 4.14) απεικονίζει τη ροή εκπαίδευσης με αλγόριθμο που παρέχεται από τον χρήστη ως Docker image. Η ροή ακολουθεί παρόμοια δομή με την προηγούμενη, με τις εξής διαφοροποιήσεις: μετά την επικύρωση, η υπηρεσία ανακτά τα μεταδεδομένα του αλγορίθμου και φορτώνει το Docker image είτε από το αποθετήριο αντικειμένων (αρχείο .tar) είτε από εξωτερικό Container Registry. Στη συνέχεια, οι υπερπαράμετροι εκπαίδευσης συγχωνεύονται (προεπιλεγμένες τιμές του αλγορίθμου σε συνδυασμό με τις τιμές που παρέχει ο χρήστης μέσω αρχείου JSON) και δημιουργείται Kubernetes Job σε πλήρως απομονωμένο περιβάλλον.



Εικόνα 4.14: Διάγραμμα Ακολουθίας - Εκπαίδευση Προσαρμοσμένου Αλγόριθμου

4.2.4.8 Εκτέλεση Πρόβλεψης

Το διάγραμμα ακολουθίας για την Εκτέλεση Μοντέλου (δείτε Εικόνα 4.15) απεικονίζει τη ροή εκτέλεσης πρόβλεψης με εκπαιδευμένο μοντέλο. Ο χρήστης μεταβαίνει στη σελίδα Executions, επιλέγει μοντέλο και σύνολο δεδομένων πρόβλεψης. Με την υποβολή, το backend επικυρώνει ότι το μοντέλο υπάρχει, ότι ο χρήστης έχει πρόσβαση σε αυτό και ότι το σύνολο δεδομένων πρόβλεψης υπάρχει σε αντίθετη περίπτωση επιστρέφεται κατάλληλο σφάλμα (404 Not Found ή 403 Forbidden). Η υπηρεσία ανακτά από το αποθετήριο αντικειμένων το εκπαιδευμένο μοντέλο και τα συνοδευτικά αρχεία, τα οποία περιλαμβάνουν τη χαρτογράφηση ετικετών (label mapping) και τη λίστα χαρακτηριστικών (feature columns) που χρησιμοποιήθηκαν κατά την εκπαίδευση. Τα αρχεία αυτά διασφαλίζουν ότι τα νέα δεδομένα υφίστανται ταυτόσημη προεπεξεργασία με αυτή της εκπαίδευσης — συγκεκριμένα, εφαρμόζονται η ίδια κωδικοποίηση κατηγορικών μεταβλητών, η ίδια επιλογή χαρακτηριστικών και η ίδια αντιστοίχιση ετικετών — ώστε να εξασφαλίζεται η συνέπεια μεταξύ εκπαίδευσης και πρόβλεψης. Στη συνέχεια, δημιουργείται Kubernetes Job πρόβλεψης και τα αποτελέσματα αποθηκεύονται σε μορφή CSV. Ο χρήστης μπορεί να τα ανακτήσει μέσω της σελίδας Executions.



Εικόνα 4.15: Διάγραμμα Ακολουθίας - Εκτέλεση Μοντέλου

4.2.5 Διάγραμμα Σχήματος Βάσης Δεδομένων

Το διάγραμμα σχήματος Βάσης Δεδομένων (δείτε Εικόνα 4.16) αποτυπώνει τη δομή της βάσης δεδομένων και τις σχέσεις μεταξύ των βασικών οντοτήτων του συστήματος.

Κεντρική θέση κατέχει η οντότητα users, η οποία συνδέεται με τους ρόλους μέσω σχέσης πολλών-προς-πολλούς. Παρότι στην τρέχουσα υλοποίηση υπάρχουν δύο ρόλοι (User, Admin), η μοντελοποίηση αυτή επιλέχθηκε συνειδητά ώστε να υποστηρίζεται η μελλοντική προσθήκη εξειδικευμένων ρόλων (π.χ. Data Scientist, ML Engineer) χωρίς αλλαγή στο σχήμα.

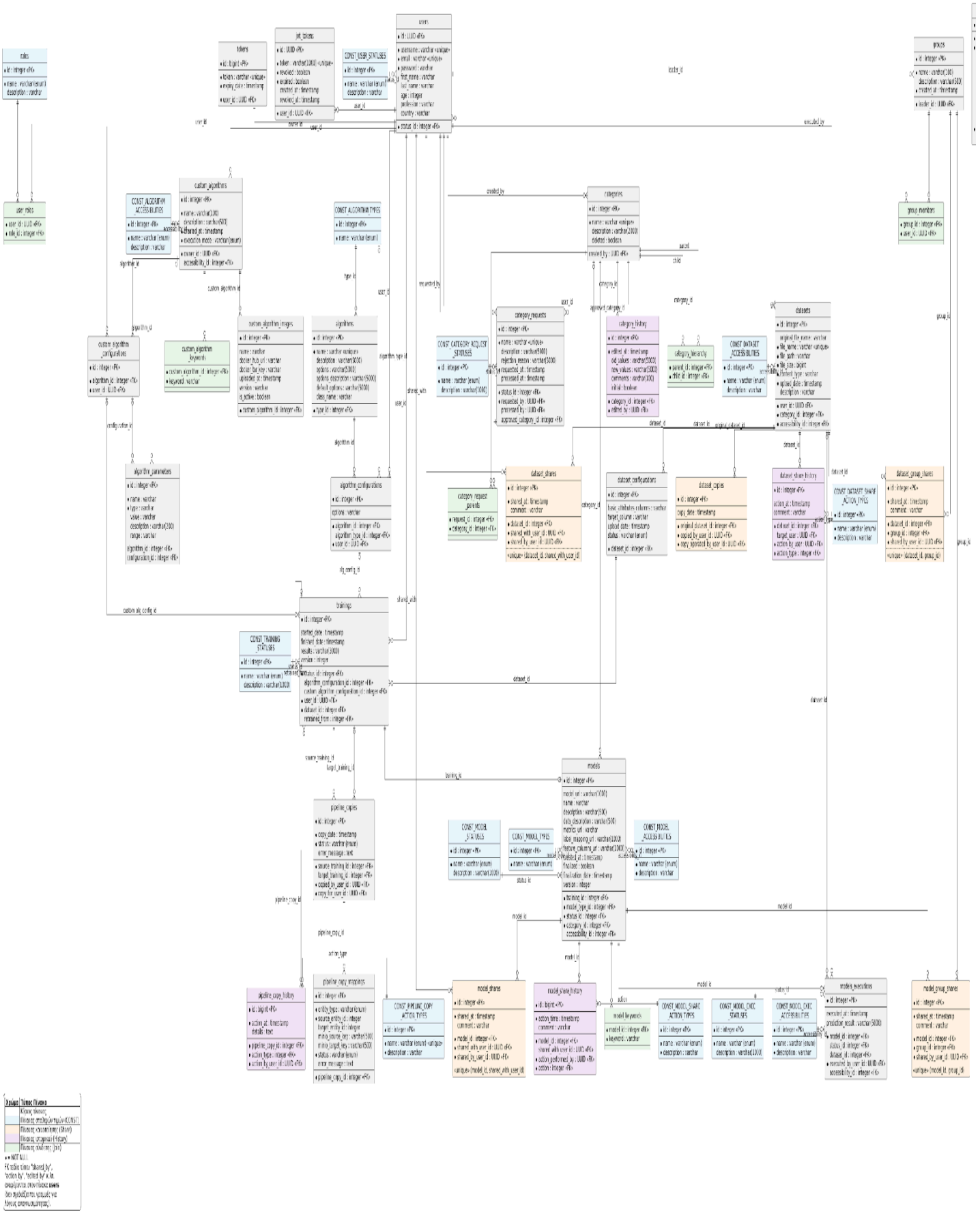
Οι χρήστες συνδέονται με datasets, trainings, προσαρμοσμένους αλγορίθμους και μοντέλα, αντικατοπτρίζοντας την ιδιοκτησία και την υπευθυνότητα για τους παραγόμενους πόρους. Ο διαμοιρασμός πόρων μοντελοποιείται μέσω των πινάκων dataset_shares και model_shares, οι οποίοι διατηρούν διπλή σχέση με τον πίνακα users: μία για τον χρήστη που πραγματοποιεί τον διαμοιρασμό (shared_by) και μία για τον χρήστη που λαμβάνει πρόσβαση (shared_with). Στις παραπάνω ενέργειες διαμοιρασμού, ο χρήστης που θα λάβει τον αντίστοιχο πόρο, έχει δικαιώματα χρήσης (read), ενώ για τα datasets προσφέρεται με τον ίδιο τρόπο η δυνατότητα αντιγραφής, όπου ουσιαστικά ο πόρος γίνεται ιδιοκτησία του χρήστη έχοντας πλήρη δικαιώματα χρήσης (read) και διαγραφής (delete).

Ιδιαίτερη σημασία δίνεται στις οντότητες που αφορούν τη ροή μηχανικής μάθησης. Κάθε training συνδέεται με το πολύ μία διαμόρφωση αλγορίθμου (algorithm_configuration ή custom_algorithm_configuration), ανάλογα με τον τύπο εκπαίδευσης, και μπορεί να παράγει το πολύ ένα model — το μοντέλο δημιουργείται μόνο με την επιτυχή ολοκλήρωση της εκπαίδευσης. Κάθε εκτέλεση πρόβλεψης (models_executions) συνδέεται τόσο με το μοντέλο που χρησιμοποιήθηκε όσο και με το σύνολο δεδομένων πρόβλεψης που χρησιμοποιείται.

Οι πίνακες αναφοράς (const_user_statuses, const_accessibility_levels, const_training_statuses, const_model_types, const_algorithm_types) χρησιμοποιούνται για την τυποποίηση καταστάσεων, τύπων και επιπέδων πρόσβασης, μειώνοντας την πιθανότητα ασυνέπειας.

Τέλος, ο πίνακας async_task_status αποτελεί σκόπιμα αποσυνδεδεμένη οντότητα: τα πεδία training_id, model_id και execution_id δεν ορίζονται ως ξένα κλειδιά, ώστε ο πίνακας να λειτουργεί ως ελαφρύς, οριζόντιος μηχανισμός παρακολούθησης εργασιών χωρίς κυκλικές εξαρτήσεις με τους πίνακες του πυρήνα. Η αποσύνδεση αυτή απλοποιεί τη

δημιουργία και εκκαθάριση εργασιών χωρίς να επηρεάζει τη συνέπεια των κύριων οντοτήτων.



Εικόνα 4.16: Διάγραμμα Σχήματος Βάσης Δεδομένων

4.3 Υλοποίηση Εφαρμογής

Η παρούσα ενότητα παρουσιάζει τις βασικές πτυχές της τεχνικής υλοποίησης της εφαρμογής. Αρχικά περιγράφονται οι τεχνολογίες και τα πλαίσια ανάπτυξης που επιλέχθηκαν, καθώς και η αιτιολόγηση των επιλογών αυτών σε σχέση με την αξιοπιστία, την επεκτασιμότητα/κλιμακωσιμότητα και τη συμβατότητα με cloud-native πρακτικές. Στη συνέχεια αναλύεται η δομή και η οργάνωση του κώδικα, με έμφαση στον διαχωρισμό ευθυνών και στη ροή δεδομένων μεταξύ των επιπέδων της εφαρμογής. Τέλος, τεκμηριώνονται οι κύριες ροές εκπαίδευσης και εκτέλεσης μοντέλων, η στρατηγική αποθήκευσης δεδομένων και αρχείων, καθώς και οι μηχανισμοί χειρισμού σφαλμάτων και ασφάλειας.

4.3.1 Τεχνολογίες και Πλαίσια Ανάπτυξης

Η υλοποίηση της εφαρμογής βασίζεται σε ένα συνεκτικό σύνολο σύγχρονων τεχνολογιών, με στόχο την παροχή μιας αξιόπιστης και επεκτάσιμης πλατφόρμας μηχανικής μάθησης, προσανατολισμένης σε περιβάλλοντα νέφους. Για το back-end επιλέχθηκε η Java 21 (LTS), αξιοποιώντας τα πλεονεκτήματα της μακροπρόθεσμης υποστήριξης, της απόδοσης και της ωριμότητας του οικοσυστήματος. Το Spring Boot (3.4.2) χρησιμοποιείται ως κύριο πλαίσιο ανάπτυξης για την υλοποίηση του RESTful API, τη διαχείριση εξαρτήσεων, καθώς και την ενσωμάτωση οριζόντιων μηχανισμών, όπως η ασφάλεια, η επικύρωση δεδομένων και η διασύνδεση με τη βάση.

Η πρόσβαση σε σχεσιακά δεδομένα και μεταδεδομένα πραγματοποιείται μέσω Spring Data JPA, το οποίο επιτρέπει τη μοντελοποίηση του πεδίου εφαρμογής με οντότητες και σχέσεις, διατηρώντας παράλληλα υψηλό επίπεδο παραγωγικότητας μέσω τυποποιημένων CRUD λειτουργιών και δηλωτικών ερωτημάτων. Η ασφάλεια υλοποιείται με Spring Security, σε συνδυασμό με JWT tokens, ώστε η αυθεντικοποίηση να είναι stateless και κατάλληλη για κατανεμημένη λειτουργία. Επιπλέον, η επικύρωση εισόδου εξασφαλίζεται με μηχανισμούς Bean Validation (Spring Boot Validation), ενώ η λειτουργία ανάκτησης κωδικού πρόσβασης υποστηρίζεται μέσω Spring Boot Mail.

Πέραν των βασικών πλαισίων του Spring, αξιοποιούνται βιβλιοθήκες τρίτων για την κάλυψη εξειδικευμένων αναγκών. Η βιβλιοθήκη JJWT χρησιμοποιείται για τη δημιουργία και επαλήθευση JWT με RSA υπογραφή, επιτυγχάνοντας σαφή διαχωρισμό μεταξύ ιδιωτικού και δημόσιου κλειδιού και αυξημένη ασφάλεια. Ως βιβλιοθήκη για την εκπαίδευση και εκτέλεση

προκαθορισμένων αλγορίθμων μηχανικής μάθησης επιλέξαμε την Weka. Η επιλογή αυτή έγινε διότι, εκτός από την συμβατότητα με την γλώσσα ανάπτυξης της εφαρμογής μας (Java), είναι μια βιβλιοθήκη που χρησιμοποιείται ευρέως σε πανεπιστήμια, για έρευνα αλλά και για εκμάθηση στα πλαίσια διδασκαλίας. Η αποθήκευση αντικειμένων (σύνολα δεδομένων, μοντέλα, μετρικές και αποτελέσματα προβλέψεων) υλοποιείται μέσω S3-compatible object storage (αποθήκευση αντικειμένων), με χρήση του MinIO Client. Για την εκτέλεση και τον έλεγχο εργασιών μηχανικής μάθησης σε περιβάλλον Kubernetes, χρησιμοποιείται ο Fabric8 Kubernetes Client, ο οποίος επιτρέπει την προγραμματιστική δημιουργία και διαχείριση Kubernetes Jobs. Παράλληλα, η βιβλιοθήκη docker-java υποστηρίζει λειτουργίες διαχείρισης Docker images (όπως φόρτωση TAR και λήψη από registry), όπου αυτό απαιτείται από τη ροή των προσαρμοσμένων αλγορίθμων. Η εξέλιξη του σχήματος της βάσης δεδομένων οργανώνεται με Flyway migrations, ώστε οι αλλαγές να είναι ελεγχόμενες, επαναλήψιμες και συμβατές με διαδικασίες συνεχούς ενσωμάτωσης.

Το front-end υλοποιείται με vanilla TypeScript και οργανώνεται ως Single Page Application (SPA) μέσω hash-based routing (δρομολόγηση με βάση κατακερματισμούς). Η επιλογή αυτή προκρίθηκε έναντι ενός πλήρους πλαισίου (framework) (όπως React, Vue ή Angular), καθώς η απαιτούμενη πολυπλοκότητα διαχείρισης κατάστασης δεν δικαιολογούσε το επιπλέον βάρος και τη δέσμευση σε συγκεκριμένο οικοσύστημα. Για τη διαδικασία ανάπτυξης και παραγωγής build χρησιμοποιείται το Vite, ένα σύγχρονο frontend build tool και development server, το οποίο χρησιμοποιείται για τη μεταγλώττιση και βελτιστοποίηση του κώδικα της διεπαφής χρήστη. Παρέχει ταχεία εκκίνηση περιβάλλοντος ανάπτυξης και μηχανισμό hot module replacement, ενώ για παραγωγικό περιβάλλον δημιουργεί βελτιστοποιημένα αρχεία μέσω μηχανισμού bundling.

Τέλος, η εφαρμογή ακολουθεί cloud-native προσέγγιση σε επίπεδο υποδομής. Όλες οι υπηρεσίες (backend, frontend) δοχειοποιούνται με Docker, ενώ το Kubernetes αξιοποιείται τόσο για την ανάπτυξη των υπηρεσιών όσο και για την εκτέλεση υπολογιστικά απαιτητικών εργασιών μηχανικής μάθησης ως απομονωμένα Jobs σε containers. Η PostgreSQL λειτουργεί ως κεντρική σχεσιακή βάση δεδομένων για μεταδεδομένα και ρυθμίσεις, ενώ το MinIO αναλαμβάνει την αποθήκευση αρχείων σε μορφή αντικειμένων. Για τοπική ανάπτυξη και δοκιμές χρησιμοποιείται το Minikube, ώστε να προσομοιώνεται ένα πραγματικό Kubernetes περιβάλλον.

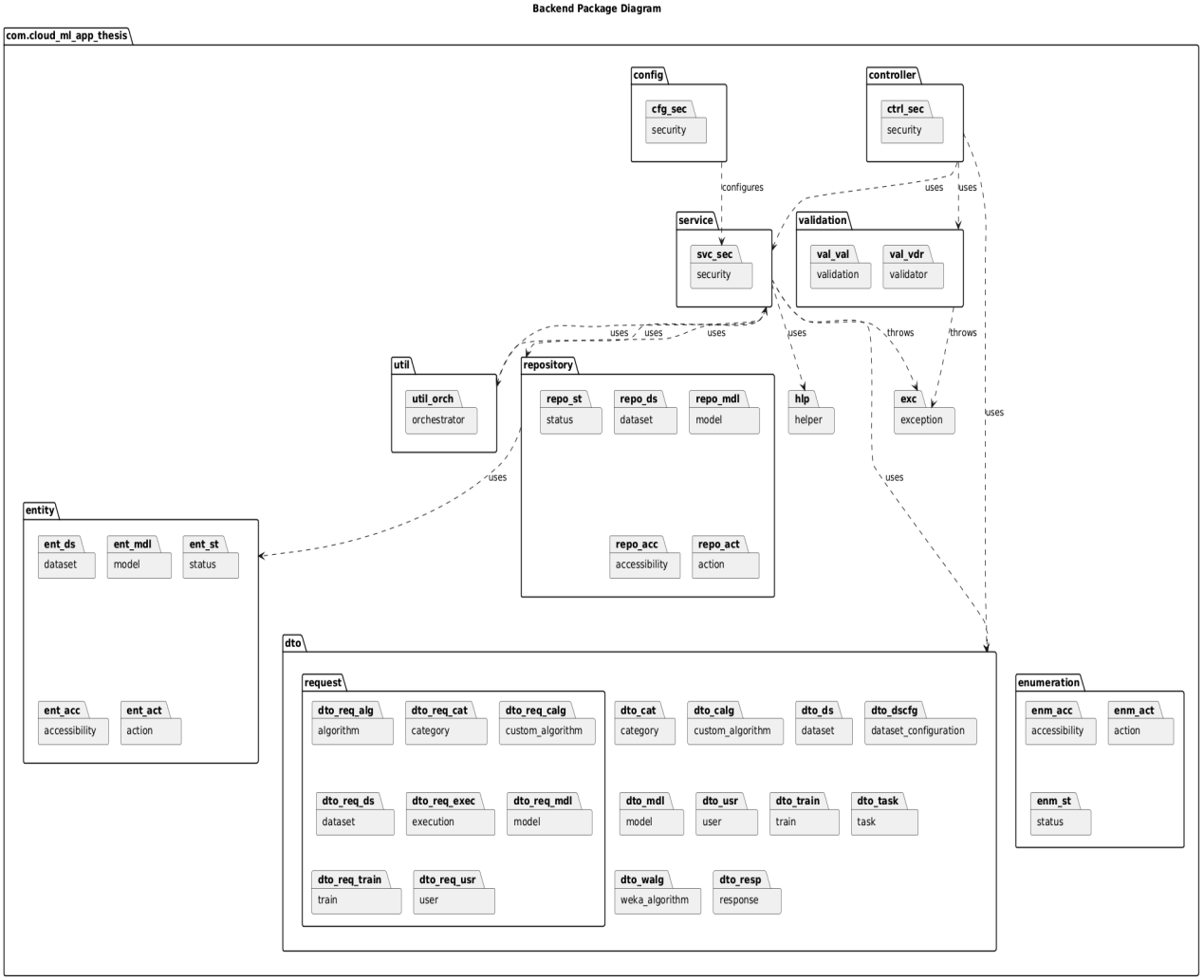
4.3.2 Δομή και Οργάνωση Κώδικα

Όπως έχει ήδη αναφερθεί, η εφαρμογή ακολουθεί αρχιτεκτονική πολλαπλών επιπέδων (layered architecture), επιδιώκοντας σαφή διαχωρισμό ευθυνών, υψηλή συνοχή εντός κάθε επιπέδου και ελαχιστοποίηση εξαρτήσεων μεταξύ διαφορετικών τμημάτων. Στο back-end, η υλοποίηση οργανώνεται σε πακέτα (δείτε Εικόνα 4.17) που αντιστοιχούν στα επίπεδα της αρχιτεκτονικής (controller, service, repository, entity), συμπληρωμένα από βοηθητικά πακέτα για τη μεταφορά δεδομένων (dto), τις καταστάσεις που μπορεί να βρεθεί κάποια οντότητα (enumeration), τη διαμόρφωση (config), τον χειρισμό σφαλμάτων (exception), την επικύρωση δεδομένων (validation) και τους υποδομικούς μηχανισμούς (util). Τα πακέτα dto, entity, enumeration και repository υποδιαιρούνται περαιτέρω ανά λειτουργικό τομέα (dataset, model, train, user, category κ.λπ.), διασφαλίζοντας ότι κάθε τομέας διατηρεί αυτόνομη δομή.

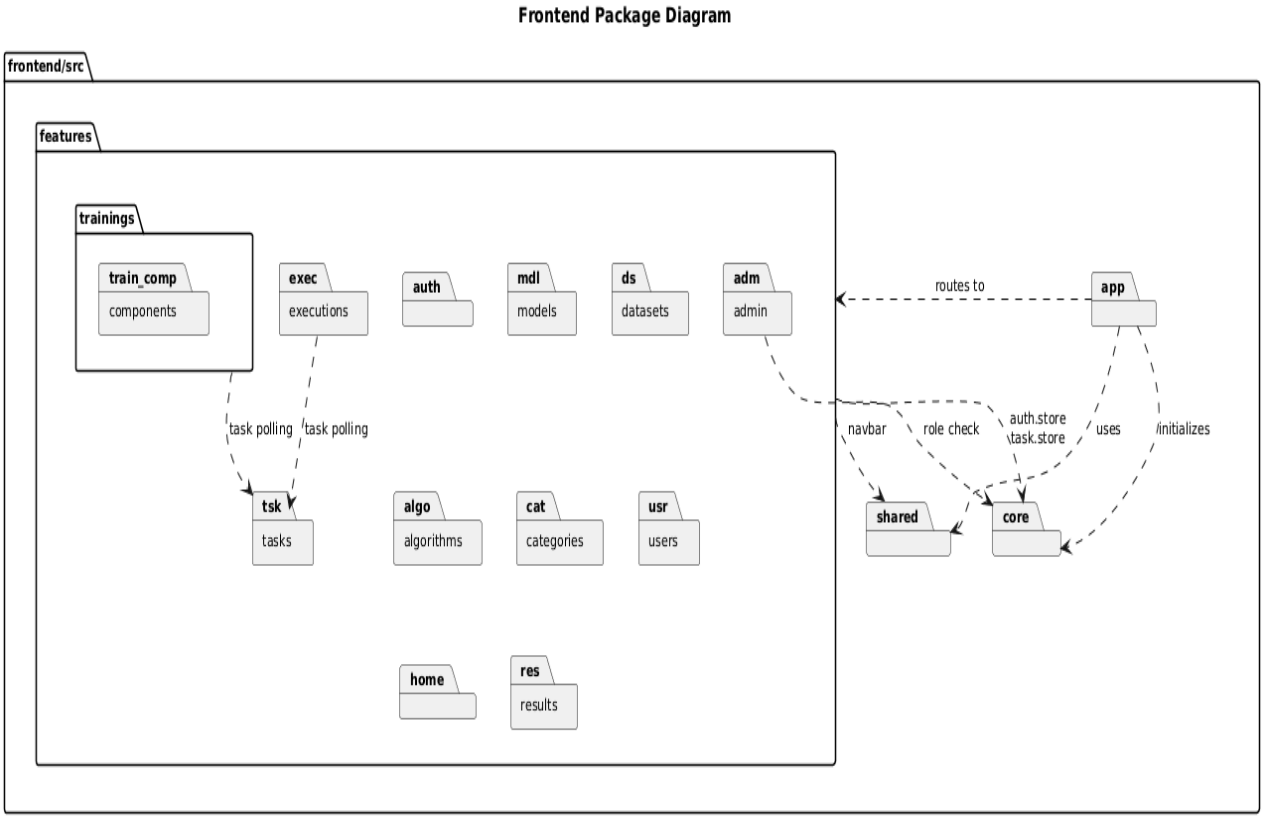
Το επίπεδο ελεγκτών (controllers) υλοποιεί τα REST endpoints και είναι υπεύθυνο για την αποδοχή των HTTP αιτημάτων, τη βασική επικύρωση εισόδου, τον έλεγχο εξουσιοδότησης και την αντιστοίχιση των δεδομένων εισόδου σε DTOs. Στη συνέχεια, η επιχειρησιακή λογική μεταφέρεται στο επίπεδο υπηρεσιών (services), το οποίο ενορχηστρώνει σύνθετες ροές, αλληλεπιδρά με αποθετήρια (repositories), υπηρεσίες αποθήκευσης αρχείων (MinIO) και μηχανισμούς εκτέλεσης containerized (δοχειοποιημένων) εργασιών μέσω μιας ενιαίας αφαίρεσης (ContainerRunner). Η πρόσβαση στη βάση δεδομένων πραγματοποιείται μέσω repository interfaces της Spring Data JPA, τα οποία παρέχουν τυποποιημένες λειτουργίες, ενώ όπου απαιτείται, χρησιμοποιούνται δηλωτικά ερωτήματα (JPQL ή native queries).

Σημαντικό σχεδιαστικό στοιχείο αποτελεί η αφαιρετική διεπαφή εκτέλεσης εργασιών (ContainerRunner), η οποία επιτρέπει εναλλακτικές υλοποιήσεις (π.χ. KubernetesJobRunner ή Docker-based runner) χωρίς μεταβολές στον κώδικα των επιχειρησιακών υπηρεσιών. Με αυτόν τον τρόπο, η πλατφόρμα διατηρεί ευελιξία ως προς το περιβάλλον εκτέλεσης, ενώ περιορίζει τη διάχυση υποδομικών λεπτομερειών στο υπόλοιπο σύστημα.

Στο front-end, ο κώδικας οργανώνεται (δείτε Εικόνα 4.18) με βάση λειτουργικές ενότητες (feature-based organization), όπου κάθε βασικός τομέας (π.χ. αυθεντικοποίηση, datasets, trainings, models, admin) υλοποιείται ως σύνολο σελίδων και επαναχρησιμοποιήσιμων components. Η διεπαφή υλοποιείται με Web Components, με κάθε σελίδα να αντιστοιχεί σε ένα custom HTML element, γεγονός που διευκολύνει την αρθρωτή ανάπτυξη και τη σταδιακή επέκταση της εφαρμογής. Παράλληλα, η επικοινωνία με το API ενοποιείται μέσω κοινών utilities (π.χ. HTTP wrapper με κεντρικό χειρισμό σφαλμάτων), ενώ το JWT token διαχειρίζεται σε ένα κεντρικό store, προκειμένου να υποστηρίζεται ομοιόμορφα η εξουσιοδότηση των κλήσεων.



Εικόνα 4.17: Διάγραμμα Πακέτων Backend



Εικόνα 4.18: Διάγραμμα Πακέτων Frontend

4.3.2.1 Διάγραμμα Κλάσεων Domain Model

Το Διάγραμμα Κλάσεων (δείτε Εικόνα 4.19) παρουσιάζει το εννοιολογικό διάγραμμα κλάσεων (conceptual class diagram) της πλατφόρμας, αποτυπώνοντας τη μοντελοποίηση του πεδίου προβλήματος (domain model) σε λογικό επίπεδο. Σκοπός του διαγράμματος δεν είναι η περιγραφή της φυσικής δομής του κώδικα, αλλά η αναγνώριση των βασικών εννοιών του πεδίου, των μεταξύ τους σχέσεων και των πολλαπλοτήτων, χωρίς την παρουσίαση χαρακτηριστικών ή μεθόδων — στοιχεία που αφορούν τα επόμενα επίπεδα αποτύπωσης (specification, implementation).

Πυρήνας του μοντέλου είναι η κλάση User, η οποία συνδέεται με σχέση πολλών-προς-πολλούς (M:N) με την κλάση Role, δεδομένου ότι ένας χρήστης δύναται να κατέχει πολλαπλούς ρόλους (π.χ. USER και ADMIN) και κάθε ρόλος αντιστοιχεί σε πολλούς χρήστες. Η πλοήγηση από τον User προς τους πόρους που κατέχει (Dataset, Training, CustomAlgorithm, ModelExecution) ορίζεται μονόδρομα (ένας-προς-πολλά), αντικατοπτρίζοντας ότι η γνώση του ιδιοκτήτη αφορά τον πόρο και όχι το αντίστροφο.

Η οντότητα Group μοντελοποιεί τις ομάδες εργασίας της πλατφόρμας. Κάθε ομάδα διαθέτει ακριβώς έναν αρχηγό (User "1" → Group "*"), οποίος μπορεί να είναι αρχηγός και σε άλλα Groups, ενώ η συμμετοχή μελών αποτυπώνεται με αμφίδρομη συσχέτιση πολλών-προς-πολλούς (User "*" — Group "*").

Η διαχείριση δεδομένων εκφράζεται μέσω των κλάσεων Dataset και DatasetConfiguration. Η μεταξύ τους σχέση είναι σύνθεση (composition): κάθε σύνολο δεδομένων περιέχει μία ή περισσότερες διαμορφώσεις, οι οποίες δεν υφίστανται ανεξάρτητα. Κάθε Dataset συνδέεται υποχρεωτικά με ακριβώς μία Category (πολλαπλότητα "*" → "1"), αποτυπώνοντας τη σχεδιαστική απόφαση ότι κάθε πόρος ανήκει σε μία μοναδική κατηγορία. Η ίδια σύμβαση ισχύει και για τα μοντέλα (Model → Category). Η κλάση Category συνδέεται αναδρομικά με τον εαυτό της σε σχέση πολλών-προς-πολλούς, καθώς η ιεραρχία κατηγοριών υλοποιείται ως κατευθυνόμενος ακυκλικός γράφος (DAG) με δυνατότητα πολλαπλών γονέων. Στο επίπεδο υλοποίησης, η κατηγορία υποστηρίζει ήπια διαγραφή (soft delete) μέσω χαρακτηριστικού κατάστασης, ώστε η αφαίρεση μιας κατηγορίας να μην προκαλεί απώλεια ιστορικών δεδομένων.

Η οντότητα CategoryRequest αποτυπώνει τη ροή έγκρισης δημιουργίας νέων κατηγοριών: κάθε αίτημα συνδέεται υποχρεωτικά με τον αιτούντα χρήστη και προαιρετικά με την εγκεκριμένη κατηγορία ("0..1"), αντιστοιχώντας στην περίπτωση εκκρεμούς ή απορριφθέντος αιτήματος.

Οι αλγόριθμοι μοντελοποιούνται σε δύο παράλληλες ιεραρχίες. Η κλάση Algorithm αντιπροσωπεύει τους προκαθορισμένους αλγορίθμους Weka και συνδέεται με πολλαπλές κλάσεις AlgorithmConfiguration. Η κλάση CustomAlgorithm αντιπροσωπεύει τους προσαρμοσμένους αλγορίθμους που δημιουργούν οι χρήστες, και περιέχει σε σύνθεση τις κλάσεις CustomAlgorithmImage (Docker image) και AlgorithmParameter. Κάθε CustomAlgorithmConfiguration περιέχει επίσης σε σύνθεση παραμέτρους (AlgorithmParameter), αποτυπώνοντας ότι η τιμή κάθε παραμέτρου είναι αδιάσπαστα συνδεδεμένη με τη συγκεκριμένη διαμόρφωση. Σημειώνεται ότι η μοντελοποίηση των παραμέτρων διαφοροποιείται μεταξύ των δύο ιεραρχιών: στους προκαθορισμένους αλγόριθμους (Algorithm/AlgorithmConfiguration), οι παράμετροι αποθηκεύονται ως πεδία κειμένου (option strings), καθώς πρόκειται για σταθερές παραμέτρους τύπου Weka που δεν απαιτούν αυτοτελή μοντελοποίηση. Αντιθέτως, στους προσαρμοσμένους αλγόριθμους, κάθε παράμετρος ορίζεται δυναμικά από τον χρήστη και απαιτεί πλουσιότερη αναπαράσταση (τύπος, τιμή, εύρος, περιγραφή), γι' αυτό μοντελοποιείται ως ξεχωριστή οντότητα (AlgorithmParameter). Σημειώνεται ότι ο τύπος αλγορίθμου (Classification, Regression, Clustering) δεν μοντελοποιείται ως ξεχωριστή κλάση αλλά ως χαρακτηριστικό

της κλάσης Algorithm στο επίπεδο υλοποίησης, καθώς πρόκειται για απλή απαρίθμηση τιμών χωρίς αυτοτελή συμπεριφορά.

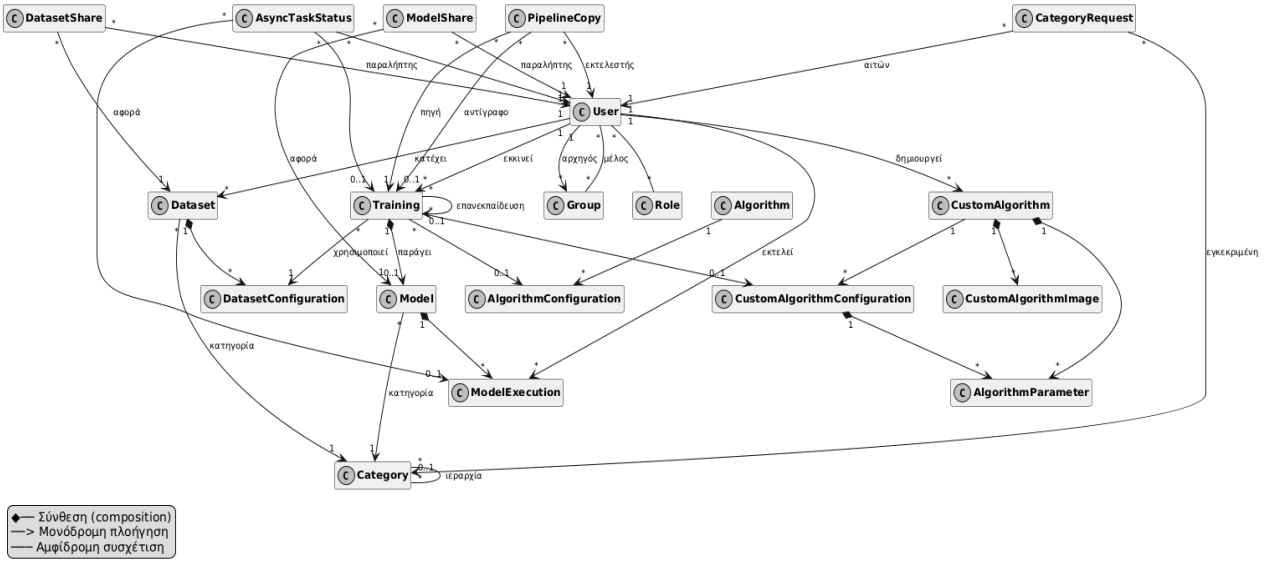
Η κλάση Training αποτελεί κεντρική οντότητα της ροής μηχανικής μάθησης. Κάθε εκπαίδευση χρησιμοποιεί ακριβώς μία DatasetConfiguration και συνδέεται προαιρετικά ("0..1") με μία AlgorithmConfiguration ή μία CustomAlgorithmConfiguration — ποτέ και με τις δύο ταυτόχρονα, αντικατοπτρίζοντας τον αμοιβαία αποκλειστικό χαρακτήρα των δύο μηχανισμών εκπαίδευσης (Weka ή BYOC). Κάθε εκπαίδευση παράγει προαιρετικά ένα Model (σύνθεση, "0..1"), καθώς η εκπαίδευση ενδέχεται να αποτύχει. Η αναδρομική σχέση Training → Training ("*" → "0..1") αποτυπώνει τη δυνατότητα επανεκπαίδευσης: πολλαπλές νέες εκπαιδεύσεις μπορούν να αναφέρουν μία προηγούμενη ως βάση.

Η κλάση Model συνδέεται σε σύνθεση με πολλαπλές εκτελέσεις ModelExecution, οι οποίες αντιπροσωπεύουν τις προβλέψεις. Στο επίπεδο υλοποίησης, κάθε εκτέλεση φέρει χαρακτηριστικό κατάστασης (status) για την παρακολούθηση του κύκλου ζωής της ασύγχρονης εργασίας.

Η κοινοποίηση πόρων μεταξύ χρηστών μοντελοποιείται μέσω των κλάσεων DatasetShare και ModelShare, καθεμία από τις οποίες συνδέεται με τον κοινοποιούμενο πόρο (σχέση «αφορά») και τον παραλήπτη. Ο ιδιοκτήτης του πόρου δεν αποθηκεύεται στις κλάσεις κοινοποίησης, καθώς προκύπτει έμμεσα μέσω των υφιστάμενων συσχετίσεων (Dataset → User, Model → Training → User). Η κλάση PipelineCopy επεκτείνει τη λειτουργικότητα κοινοποίησης, αποτυπώνοντας την αντιγραφή ολόκληρου pipeline εκπαίδευσης — η οποία περιλαμβάνει το dataset, τη διαμόρφωση αλγορίθμου, την εκπαίδευση και το παραγόμενο μοντέλο μαζί με όλα τα αρχεία του (μετρικές, label mapping, feature columns). Συνδέεται με την πηγαία εκπαίδευση, προαιρετικά με την εκπαίδευση-αντίγραφο, και με τον εκτελεστή της αντιγραφής.

Τέλος, η κλάση AsyncTaskStatus αποτυπώνει τη γενική οντότητα παρακολούθησης ασύγχρονων εργασιών της πλατφόρμας. Συνδέεται υποχρεωτικά με τον χρήστη που ξεκίνησε την εργασία και προαιρετικά με μία εκπαίδευση ή μία εκτέλεση πρόβλεψης, ανάλογα με τον τύπο της εργασίας.

Στο υπόμνημα του διαγράμματος διακρίνονται τρεις τύποι σχέσεων: η σύνθεση (◆) υποδηλώνει ότι το περιεχόμενο μέρος δεν υφίσταται χωρίς το σύνολο, η μονόδρομη πλοήγηση (→) ορίζει κατεύθυνση γνώσης, και η αμφίδρομη συσχέτιση (—) επιτρέπει πλοήγηση και προς τις δύο κατευθύνσεις.



Εικόνα 4.19: Διάγραμμα Κλάσεων

4.3.3 Υλοποίηση Ροών Εκπαίδευσης και Εκτέλεσης Μοντέλων

Οι ροές εκπαίδευσης και εκτέλεσης μοντέλων, των οποίων η σχεδίαση παρουσιάστηκε μέσω διαγραμμάτων ακολουθίας στην Ενότητα 4.2.4, υλοποιούνται με κοινή υποδομή ασύγχρονης εκτέλεσης σε απομονωμένα containers. Η παρούσα ενότητα εστιάζει στις λεπτομέρειες υλοποίησης που δεν αποτυπώνονται στα διαγράμματα.

4.3.3.1 Μηχανισμός Εκτέλεσης Εργασιών (ContainerJobRunner)

Ο ContainerJobRunner αποτελεί τον συνδετικό κρίκο μεταξύ της εφαρμογής και του Kubernetes cluster. Κάθε εργασία δημιουργείται ως Kubernetes Job με τα ακόλουθα χαρακτηριστικά:

- Απενεργοποίηση retries (backoffLimit: 0), ώστε αποτυχημένες εκτελέσεις να μην επαναλαμβάνονται αυτόματα και να αναφέρονται άμεσα στον χρήστη.
- Πολιτική επανεκκίνησης Never, ώστε τα pods να μην επανεκκινούνται σε περίπτωση σφάλματος.
- Χρονικό όριο αυτόματης εκκαθάρισης (TTL seconds after finished), ώστε τα ολοκληρωμένα Jobs να διαγράφονται αυτόματα μετά από ορισμένο χρόνο, απελευθερώνοντας πόρους του cluster.

- Περιορισμοί πόρων (memory/CPU limits), ώστε μία εργασία να μην μπορεί να καταναλώσει υπερβολικούς πόρους εις βάρος άλλων.
- Απομόνωση ασφαλείας: τα containers εκτελούνται χωρίς δικαιώματα root και χωρίς πρόσβαση στο δίκτυο του host.

Η ακύρωση εργασίας υλοποιείται μέσω διαγραφής του Job με πολιτική foreground propagation, η οποία διασφαλίζει τη διαγραφή των συναφών Pods και την άμεση παύση κατανάλωσης πόρων.

4.3.3.2 Ανταλλαγή Δεδομένων μεταξύ Backend και Jobs

Η επικοινωνία μεταξύ backend και containers πραγματοποιείται μέσω κοινόχρηστου PersistentVolumeClaim (PVC), το οποίο προσαρτάται τόσο στο backend όσο και στα Jobs. Για κάθε εργασία δημιουργείται μοναδικός υποφάκελος που περιέχει:

- Κατάλογος /data: το σύνολο δεδομένων, το αρχείο παραμέτρων (params.json) και, στην περίπτωση προσαρμοσμένων αλγορίθμων, τα scripts εκτέλεσης.
- Κατάλογος /model: χώρος εξόδου για το παραγόμενο μοντέλο, τις μετρικές αξιολόγησης και τα αποτελέσματα πρόβλεψης.

Μετά την ολοκλήρωση της εργασίας, το backend ανακτά τα αρχεία εξόδου, τα αποθηκεύει στο object storage και εκκαθαρίζει τον προσωρινό φάκελο.

4.3.3.3 Μηχανισμός Μετασχηματισμού Δεδομένων

Για την υποστήριξη προσαρμοσμένων αλγορίθμων χωρίς να μεταφέρεται πολυπλοκότητα στον χρήστη, υλοποιείται μηχανισμός αυτόματης προεπεξεργασίας δεδομένων. Κατά την εκπαίδευση, η πλατφόρμα εφαρμόζει:

- One-hot encoding για κατηγορικά χαρακτηριστικά εισόδου (features), μετατρέποντάς τα σε αριθμητικές αναπαραστάσεις.
- Label encoding για τη μεταβλητή-στόχο (target), αντιστοιχίζοντας κάθε ετικέτα σε ακέραιο αριθμό.

Παράλληλα, αποθηκεύονται βοηθητικά artifacts:

- label_mapping.json: η αντιστοίχιση μεταξύ αριθμητικών τιμών και αρχικών ετικετών.
- feature_columns.json: η λίστα και η σειρά των χαρακτηριστικών μετά τον μετασχηματισμό.

Τα artifacts αυτά χρησιμοποιούνται κατά τη φάση πρόβλεψης ώστε να εφαρμοστεί ταυτόσημη προεπεξεργασία στα νέα δεδομένα και να αποκωδικοποιηθούν τα αποτελέσματα στις αρχικές ετικέτες, διασφαλίζοντας συνέπεια μεταξύ εκπαίδευσης και πρόβλεψης.

4.3.3.4 Υλοποίηση Ροής Πρόβλεψης

Η ροή πρόβλεψης ακολουθεί ανάλογη υποδομή με την εκπαίδευση, με τις εξής διαφοροποιήσεις σε επίπεδο υλοποίησης:

- Ανάκτηση του εκπαιδευμένου μοντέλου και των βοηθητικών artifacts (label mapping, feature columns) από το object storage, αντί για σύνολο δεδομένων εκπαίδευσης.
- Δημιουργία δύο καταλόγων στον κοινόχρηστο χώρο: /data (νέα δεδομένα πρόβλεψης) και /model (μοντέλο και artifacts).
- Εφαρμογή του ίδιου μετασχηματισμού στα νέα δεδομένα βάσει των αποθηκευμένων artifacts, ώστε η μορφή εισόδου να αντιστοιχεί στη μορφή που αναμένει το μοντέλο.
- Αποκωδικοποίηση αριθμητικών προβλέψεων στις αρχικές ετικέτες μέσω του label mapping.
- Αποθήκευση αποτελεσμάτων σε μορφή CSV στο object storage, ώστε ο χρήστης να μπορεί να τα ανακτήσει μέσω της εφαρμογής.

Η κοινή στρατηγική μετασχηματισμών μεταξύ εκπαίδευσης και πρόβλεψης αποτελεί κρίσιμο στοιχείο υλοποίησης, καθώς μειώνει τις ασυμβατότητες σχήματος και εξαλείφει μια συχνή πηγή σφαλμάτων σε συστήματα μηχανικής μάθησης.

4.3.4 Διαχείριση Αποθήκευσης και Δεδομένων

Η πλατφόρμα υιοθετεί διπλή στρατηγική αποθήκευσης, αξιοποιώντας αφενός σχεσιακή βάση δεδομένων για μεταδεδομένα και ρυθμίσεις και αφετέρου object storage για αρχεία μεγάλου όγκου και παραγόμενα artifacts (τεχνήματα). Η προσέγγιση αυτή διαχωρίζει τον δομημένο επιχειρησιακό πυρήνα (οντότητες όπως χρήστες, trainings, μοντέλα, ρυθμίσεις) από τα μη δομημένα ή ημι-δομημένα δεδομένα (σύνολα δεδομένων, μοντέλα ως αρχεία,

μετρικές, αποτελέσματα προβλέψεων), επιτυγχάνοντας καλύτερη κλιμάκωση και ευκολότερη διαχείριση.

Για την αποθήκευση αρχείων χρησιμοποιείται το MinIO, ως S3-compatible object storage. Η σχετική λειτουργικότητα συγκεντρώνεται σε μια εξειδικευμένη υπηρεσία (ObjectStorageService), η οποία παρέχει ενιαίο σημείο πρόσβασης για upload και download, με ελέγχους εγκυρότητας (π.χ. επιβεβαίωση επιτρεπόμενων buckets) και υποστήριξη αποδοτικών ροών μεταφοράς δεδομένων. Παράλληλα, υιοθετείται σαφής και προβλέψιμη δομή κλειδιών (keys) ανά bucket, ώστε να διευκολύνεται η ιχνηλασιμότητα των artifacts, η συσχέτισή τους με τον χρήστη και τον χρόνο δημιουργίας, καθώς και η μετέπειτα ανάκτηση και καθαρισμός τους.

Η πρόσβαση στη βάση δεδομένων υλοποιείται μέσω Spring Data JPA repositories. Με τον τρόπο αυτό, οι βασικές λειτουργίες ανάκτησης και αποθήκευσης δεδομένων υλοποιούνται δηλωτικά, ενώ σε περιπτώσεις που απαιτούνται πιο εξειδικευμένα κριτήρια, χρησιμοποιούνται custom queries. Η προσέγγιση αυτή επιτρέπει την εξέλιξη του σχήματος και της επιχειρησιακής λογικής με ελεγχόμενο τρόπο, σε συνδυασμό με τις μεταναστεύσεις της βάσης (Flyway).

4.3.5 Χειρισμός Σφαλμάτων και Ζητήματα Ασφάλειας

Η εφαρμογή υλοποιεί κεντρικοποιημένο χειρισμό σφαλμάτων, με στόχο την ομοιόμορφη και αυτο-επεξηγούμενη συμπεριφορά του API προς τον πελάτη, καθώς και την αποφυγή διαρροής τεχνικών λεπτομερειών. Ο μηχανισμός αυτός βασίζεται σε global exception handling (καθολικός χειρισμός εξαιρέσεων), όπου εξαιρέσεις, όπως σφάλματα επικύρωσης εισόδου, περιπτώσεις μη εύρεσης πόρων, μη εξουσιοδοτημένη πρόσβαση και συγκρούσεις λόγω ταυτόχρονων ενημερώσεων, μεταφράζονται σε κατάλληλους HTTP κωδικούς (400, 404, 403, 409 αντίστοιχα). Οι απαντήσεις του API ακολουθούν τυποποιημένο σχήμα απόκρισης (GenericResponse), ώστε ο πελάτης να μπορεί να ερμηνεύει σταθερά τόσο τις επιτυχείς απαντήσεις όσο και τα σφάλματα, αξιοποιώντας κωδικούς σφάλματος και μηνύματα προσανατολισμένα στον χρήστη.

Σε επίπεδο ασφάλειας, η αυθεντικοποίηση βασίζεται σε JWT tokens, τα οποία υπογράφονται με RSA, επιτρέποντας επαλήθευση με δημόσιο κλειδί και περιορίζοντας την έκθεση ευαίσθητου υλικού. Η εξουσιοδότηση υλοποιείται με role-based access control (RBAC) (έλεγχος πρόσβασης με βάση ρόλους), όπου οι ρόλοι μεταφέρονται ως claims στο token και επιβάλλονται μέσω φίλτρων ασφαλείας και δηλωτικών ελέγχων πρόσβασης (π.χ. με

annotations). Η προσέγγιση stateless ασφάλειας είναι ιδιαίτερα κατάλληλη για περιβάλλον Kubernetes, καθώς υποστηρίζει πολλαπλά replicas και οριζόντια κλιμάκωση χωρίς ανάγκη για server-side sessions.

Ιδιαίτερη μέριμνα δόθηκε στον ασφαλή χειρισμό ακύρωσης χρονοβόρων εργασιών (training/prediction), έτσι ώστε ο χρήστης να μπορεί να ακυρώσει την διεργασία σε οποιαδήποτε φάση που προηγείται της ολοκλήρωσης. Η ακύρωση υλοποιείται ως συνδυασμός λογικής σήμανσης (stop request) και υποδομικής παύσης (διαγραφή Kubernetes Job), ώστε να αποφεύγεται η άσκοπη κατανάλωση πόρων και να διασφαλίζεται ότι δεν παράγονται ενδιάμεσα ή ασυνεπή αποτελέσματα. Παράλληλα, εφαρμόζεται στρατηγική καθαρισμού πόρων (cleanup) σε επίπεδο υποδομής και αποθήκευσης, συμπεριλαμβανομένης της διαγραφής Jobs/Pods, της απομάκρυνσης προσωρινών αρχείων από shared volumes και, όπου απαιτείται, της αποκατάστασης ή σήμανσης της κατάστασης σχετικών οντοτήτων στη βάση, ώστε το σύστημα να παραμένει συνεπές μετά από διακοπή ή αποτυχία.

4.4 Δοκιμές

Η ανάπτυξη της εφαρμογής συνοδεύτηκε από συστηματική διαδικασία δοκιμών με στόχο τη διασφάλιση της ποιότητας, της αξιοπιστίας και της ορθής λειτουργίας του συστήματος. Η στρατηγική δοκιμών που ακολουθήθηκε καλύπτει πολλαπλά επίπεδα — μοναδιαίες δοκιμές (unit tests), δοκιμές ενοποίησης (integration tests), δοκιμές συστήματος (system tests) και χειρωνακτικές λειτουργικές δοκιμές — ώστε να ελεγχθούν τόσο οι μεμονωμένες μονάδες λογισμικού όσο και η ορθή συνεργασία των επιμέρους υποσυστημάτων/συστατικών. Ιδιαίτερη έμφαση δόθηκε στη δοκιμή σεναρίων αποτυχίας, προκειμένου να αξιολογηθεί η ανθεκτικότητα της εφαρμογής και η αποτελεσματικότητα των μηχανισμών χειρισμού σφαλμάτων.

Ως πλαίσιο αυτοματοποιημένων δοκιμών χρησιμοποιήθηκε το JUnit 5 (Jupiter) σε συνδυασμό με τα Spring Boot Test, Mockito 5.12.0 και REST Assured 5.4.0. Οι μοναδιαίες δοκιμές αυτοματοποιήθηκαν πλήρως μέσω κώδικα, ενώ οι χειρωνακτικές λειτουργικές δοκιμές εκτελέστηκαν μέσω του εργαλείου Postman για τα REST API endpoints και του εργαλείου γραμμής εντολών kubectl για τη δοκιμή εντός του Kubernetes cluster.

Μοναδιαίες δοκιμές (Unit Tests). Οι μοναδιαίες δοκιμές (14 αρχεία δοκιμών, 200+ σενάρια) υλοποιήθηκαν με κώδικα χρησιμοποιώντας JUnit 5 και τη βιβλιοθήκη Mockito

5.12.0 για την απομόνωση κάθε μονάδας από τις εξωτερικές της εξαρτήσεις. Για τη δοκιμή των REST controllers χρησιμοποιήθηκε η βιβλιοθήκη MockMvc του Spring Test. Οι δοκιμές εστίασαν στην επιχειρησιακή λογική των υπηρεσιών, χωρίς προσπέλαση στη βάση δεδομένων ή σε εξωτερικά συστήματα, επιτρέποντας ταχεία εκτέλεση και ακριβή εντοπισμό σφαλμάτων. Στο πλαίσιο αυτό δοκιμάστηκαν η διαχείριση χρηστών (εγγραφή, αυθεντικοποίηση, αλλαγή κωδικού, διαγραφή λογαριασμού), η διαχείριση κατηγοριών μοντέλων με ροή έγκρισης, η δημιουργία και αναζήτηση αλγορίθμων (Weka και προσαρμοσμένων), η διαχείριση κατάστασης μοντέλων, η κοινοποίηση μοντέλων και συνόλων δεδομένων σε ομάδες χρηστών, η αντιγραφή pipelines, καθώς και οι λειτουργίες αποθήκευσης και ανάκτησης αντικειμένων στο MinIO. Εκτενώς δοκιμάστηκαν επίσης τα σενάρια αποτυχίας, όπως η πρόσβαση χωρίς εξουσιοδότηση, η αναζήτηση ανύπαρκτων οντοτήτων και η εισαγωγή μη έγκυρων δεδομένων.

Δοκιμές ενοποίησης (Integration Tests). Σε δεύτερο επίπεδο, υλοποιήθηκαν αυτοματοποιημένες δοκιμές ενοποίησης (5 αρχεία δοκιμών) με JUnit 5 και τον μηχανισμό @SpringBootTest, οι οποίες φόρτωναν τον πλήρη application context του Spring Boot και εκτελούσαν λειτουργίες απευθείας στο επίπεδο υπηρεσιών (service layer) με σύνδεση σε πραγματικά υποσυστήματα. Συγκεκριμένα, οι δοκιμές αυτές χρησιμοποιούσαν τη σχεσιακή βάση δεδομένων PostgreSQL, το σύστημα αποθήκευσης αντικειμένων MinIO (συμβατό με S3) και τα περιβάλλοντα εκτέλεσης containers — ήτοι τον Docker Engine (μέσω της βιβλιοθήκης Docker Java 3.3.6) και το Kubernetes/Minikube (μέσω του Fabric8 Kubernetes Client 6.13.1). Οι δοκιμές αυτές κάλυψαν τις ροές εκπαίδευσης και πρόβλεψης τόσο με αλγορίθμους Weka (containerized εκτέλεση μέσω Weka Runner) όσο και με προσαρμοσμένους αλγορίθμους (custom Docker containers), επαληθεύοντας τη σωστή χρήση παραμέτρων εκπαίδευσης, τη δημιουργία Kubernetes Jobs, τη μεταφορά δεδομένων μέσω κοινών volumes, και την παραγωγή των αναμενόμενων artifacts (αρχεία μοντέλων, μετρικές αξιολόγησης, αποτελέσματα προβλέψεων). Επίσης, δοκιμάστηκε αυτοματοποιημένα η δημιουργία προσαρμοσμένων αλγορίθμων (custom algorithms) με μεταφόρτωση Docker image και αρχείου παραμέτρων.

Δοκιμές συστήματος (System Tests). Σε τρίτο επίπεδο, υλοποιήθηκαν αυτοματοποιημένες δοκιμές ενοποίησης (2 αρχεία δοκιμών) με τη βιβλιοθήκη REST Assured 5.4.0, οι οποίες δοκίμαζαν την εφαρμογή μέσω κλήσεων HTTP στο REST API, προσομοιώνοντας τη συμπεριφορά πραγματικού πελάτη. Οι δοκιμές αυτές κάλυπταν πλήρεις ροές εργασίας: αυθεντικοποίηση, μεταφόρτωση συνόλου δεδομένων, εκκίνηση εκπαίδευσης, αναμονή ολοκλήρωσης εργασίας, οριστικοποίηση μοντέλου, εκτέλεση πρόβλεψης και ανάκτηση αποτελεσμάτων — τόσο για τη ροή αλγορίθμων Weka όσο και για

τη ροή προσαρμοσμένων αλγορίθμων. Σε αντίθεση με τις δοκιμές ενοποίησης που εκτελούσαν υπηρεσίες απευθείας, οι δοκιμές συστήματος διέσχιζαν σχεδόν όλα τα αρχιτεκτονικά επίπεδα (HTTP → Controller → Service → Repository → Database/MinIO/Kubernetes), επιβεβαιώνοντας ότι η εφαρμογή λειτουργεί σωστά ως ολοκληρωμένο σύστημα.

Χειρωνακτικές λειτουργικές δοκιμές. Πέραν των αυτοματοποιημένων δοκιμών, εκτελέστηκαν χειρωνακτικές λειτουργικές δοκιμές χρησιμοποιώντας το εργαλείο Postman για την αποστολή HTTP αιτημάτων στα REST API endpoints και τον έλεγχο των απαντήσεων. Μέσω Postman δοκιμάστηκαν οι πλήρεις ροές χρήσης, συμπεριλαμβανομένης της αυθεντικοποίησης, της διαχείρισης συνόλων δεδομένων, της εκπαίδευσης μοντέλων, της εκτέλεσης προβλέψεων, και ειδικότερα η λειτουργία BYOC (Bring Your Own Container) για εκτέλεση αλγορίθμων σε οποιαδήποτε γλώσσα προγραμματισμού.

Παράλληλα, χρησιμοποιήθηκε το εργαλείο γραμμής εντολών kubectl για τη δοκιμή της εφαρμογής εντός του Kubernetes cluster, τον έλεγχο της κατάστασης των Pods και Jobs, καθώς και την επαλήθευση της ορθής λειτουργίας σε περιβάλλον ενορχήστρωσης containers.

Συνολικά, η διαδικασία δοκιμών κάλυψε πολλαπλά επίπεδα ελέγχου — από τη μεμονωμένη μονάδα λογισμικού μέχρι το ολοκληρωμένο υπο-σύστημα του backend — και συνέβαλε καθοριστικά στη διασφάλιση της ασφάλειας, της ακρίβειας και της επαναληψιμότητας των βασικών ροών της εφαρμογής.

4.5 Ικανοποίηση Απαιτήσεων

Η παρούσα ενότητα αξιολογεί τον βαθμό στον οποίο η υλοποιημένη εφαρμογή ικανοποιεί τις λειτουργικές και μη λειτουργικές απαιτήσεις που προσδιορίστηκαν στην ενότητα 4.1. Η αξιολόγηση δεν περιορίζεται στη διαπίστωση της ύπαρξης ή μη συγκεκριμένων λειτουργιών, αλλά λαμβάνει υπόψη τη συνολική συμπεριφορά του συστήματος, τις αρχιτεκτονικές και σχεδιαστικές επιλογές που υιοθετήθηκαν, καθώς και τα αποτελέσματα των δοκιμών που παρουσιάστηκαν στην προηγούμενη ενότητα.

Η ικανοποίηση των απαιτήσεων εξετάζεται υπό το πρίσμα της πρακτικής χρηστικότητας, της ασφάλειας, της επεκτασιμότητας και της αξιοπιστίας του συστήματος, σε συνάρτηση με τους περιορισμούς που τέθηκαν κατά τον σχεδιασμό και την υλοποίηση της εφαρμογής στο πλαίσιο της παρούσας διπλωματικής εργασίας.

4.5.1 Μεθοδολογία Αξιολόγησης Απαιτήσεων

Η αξιολόγηση των απαιτήσεων βασίζεται σε δύο άξονες:

1. Λειτουργική κάλυψη, δηλαδή τον βαθμό στον οποίο η εφαρμογή παρέχει τη ζητούμενη λειτουργικότητα σύμφωνα με τις αρχικές προδιαγραφές.
2. Μη λειτουργική κάλυψη, δηλαδή τον βαθμό ικανοποίησης των μη λειτουργικών (ποιοτικών) απαιτήσεων, όπως η ασφάλεια, η απόδοση, η κλιμάκωση και η αξιοπιστία, οι οποίες επηρεάζουν τη συνολική εμπειρία χρήσης και τη βιωσιμότητα της εφαρμογής.

Για κάθε απαίτηση καταγράφεται ο βαθμός ικανοποίησής της ως πλήρης ή μερικός, συνοδευόμενος από σύντομη τεκμηρίωση που αιτιολογεί την αντίστοιχη αξιολόγηση. Όπου ο βαθμός κάλυψης είναι μερικός, η αιτιολόγηση στηρίζεται στις σχεδιαστικές αποφάσεις που ελήφθησαν κατά την ανάπτυξη, με στόχο τη διατήρηση της απλότητας, της συντηρησιμότητας και της καθαρής αρχιτεκτονικής.

4.5.2 Ανάλυση Πλήρους Ικανοποίησης Απαιτήσεων

Η πλειονότητα των λειτουργικών απαιτήσεων ικανοποιείται πλήρως. Συγκεκριμένα, λειτουργίες που αφορούν τη διαχείριση χρηστών, τη διαχείριση συνόλων δεδομένων, την εκπαίδευση μοντέλων, την εκτέλεση προβλέψεων και την υποστήριξη προσαρμοσμένων αλγορίθμων υλοποιούνται με τρόπο που ανταποκρίνεται πλήρως στις αρχικές προδιαγραφές.

Ιδιαίτερη έμφαση δόθηκε στην ασύγχρονη εκτέλεση υπολογιστικά απαιτητικών διεργασιών, επιτρέποντας την αποσύνδεση της αλληλεπίδρασης του χρήστη από τη διάρκεια εκτέλεσης των αλγορίθμων. Παράλληλα, οι μη λειτουργικές απαιτήσεις που σχετίζονται με την ασφάλεια, τη διαλειτουργικότητα και την κλιμάκωση καλύπτονται επαρκώς μέσω της χρήσης τεχνολογιών νέφους, δοχειοποίησης και σαφώς ορισμένων διεπαφών.

4.5.3 Ικανοποίηση Λειτουργικών Απαιτήσεων

Η παρούσα ενότητα αξιολογεί τον βαθμό στον οποίο η υλοποιημένη εφαρμογή ικανοποιεί τις λειτουργικές απαιτήσεις που τέθηκαν κατά το στάδιο ανάλυσης. Η αξιολόγηση βασίζεται

στη λειτουργική συμπεριφορά του συστήματος και στις σχεδιαστικές αποφάσεις που καθόρισαν τον τρόπο υλοποίησής τους.

4.5.3.1 Διαχείριση Χρηστών

Η εφαρμογή υποστηρίζει πλήρως τη διαχείριση χρηστών, καλύπτοντας όλες τις απαιτήσεις που σχετίζονται με τον κύκλο ζωής ενός λογαριασμού (χρήστη). Παρέχεται μηχανισμός εγγραφής χρήστη με υποχρεωτική εισαγωγή βασικών στοιχείων (ονοματεπώνυμο, email και κωδικός), καθώς και προαιρετικών πληροφοριών (ηλικία, επάγγελμα και χώρα διαμονής). Κατά την εγγραφή, εφαρμόζονται έλεγχοι εγκυρότητας τόσο στο email όσο και στον κωδικό πρόσβασης, ο οποίος απαιτείται να ακολουθεί προκαθορισμένο μοτίβο ασφάλειας (security pattern).

Η αυθεντικοποίηση υλοποιείται μέσω μηχανισμού έκδοσης διακριτικών πρόσβασης (JWT), τα οποία επιτρέπουν την ασφαλή πρόσβαση σε προστατευμένες λειτουργίες της εφαρμογής. Υποστηρίζονται λειτουργίες τροποποίησης στοιχείων χρήστη, αλλαγής κωδικού με επαλήθευση του παλαιού, καθώς και επαναφοράς κωδικού μέσω ηλεκτρονικού ταχυδρομείου. Επιπλέον, παρέχεται δυνατότητα διαγραφής λογαριασμού τόσο από τον ίδιο τον χρήστη όσο και από διαχειριστή, καλύπτοντας πλήρως τις αντίστοιχες απαιτήσεις.

Η ύπαρξη διακριτών ρόλων (απλός χρήστης και διαχειριστής) επιτρέπει την εφαρμογή διαφορετικών πολιτικών πρόσβασης και διαχείρισης, με τον διαχειριστή να έχει αυξημένα δικαιώματα εποπτείας. Δηλαδή ο διαχειριστής έχει πλήρη δικαιώματα ως προς την διαχείριση των χρηστών, των δεδομένων (datasets), των εκπαιδεύσεων και των μεταδεδομένων τους, των μοντέλων, των προβλέψεων και των μεταδεδομένων τους, των πληροφοριών των προκαθορισμένων αλγορίθμων αλλά και των δεδομένων που αφορούν όλα τα status των οντοτήτων της εφαρμογής. Μάλιστα ο διαχειριστής μπορεί να διαχειρίζεται όλα τα δεδομένα των άλλων χρηστών καθώς και τους λογαριασμούς τους.

4.5.3.2 Διαχείριση Εκπαιδεύσεων Αλγορίθμων Μηχανικής Μάθησης

Η εφαρμογή καλύπτει πλήρως τις απαιτήσεις που σχετίζονται με την εκπαίδευση αλγορίθμων μηχανικής μάθησης. Ο χρήστης μπορεί να αιτηθεί την εκπαίδευση αλγορίθμου (είτε προκαθορισμένου είτε custom) παρέχοντας σύνολο δεδομένων σε κοινές μορφές (CSV, Excel, ARFF), ορίζοντας τα βασικά χαρακτηριστικά και τη στήλη πρόβλεψης. Παρέχεται

ευελιξία ως προς τη διαμόρφωση παραμέτρων του αλγορίθμου, με δυνατότητα χρήσης προκαθορισμένων ή προσαρμοσμένων τιμών.

Η εκπαίδευση υλοποιείται με ασύγχρονο τρόπο, δεδομένου ότι πρόκειται για υπολογιστικά απαιτητική διαδικασία. Με την υποβολή του αιτήματος, το σύστημα επιστρέφει μοναδικό αναγνωριστικό εκπαίδευσης, μέσω του οποίου ο χρήστης μπορεί να παρακολουθεί την κατάστασή της (προς εκκίνηση, σε εξέλιξη, ολοκληρωμένη, αποτυχημένη). Σε περιπτώσεις αποτυχίας, καταγράφονται και διατίθενται πληροφορίες σφάλματος για διευκόλυνση της διάγνωσης.

Υποστηρίζονται επιπλέον λειτουργίες εκφόρτωσης των αποτελεσμάτων εκπαίδευσης, εμφάνισης και φιλτραρίσματος εκπαιδεύσεων, ρητής παύσης εκπαίδευσης και επανεκπαίδευσης με βάση υπάρχουσα διαμόρφωση, καλύπτοντας πλήρως τις σχετικές λειτουργικές απαιτήσεις.

4.5.3.3 Διαχείριση Μοντέλων Μηχανικής Μάθησης

Η εφαρμογή διαχωρίζει εννοιολογικά την έννοια της εκπαίδευσης από αυτήν του μοντέλου μηχανικής μάθησης, ικανοποιώντας τις αντίστοιχες απαιτήσεις σχεδιασμού. Ένα μοντέλο δημιουργείται ως αποτέλεσμα επιτυχημένης εκπαίδευσης και συνοδεύεται από πλούσιο σύνολο μεταδεδομένων, όπως όνομα, περιγραφή, κατηγορία, λέξεις-κλειδιά και επίπεδο πρόσβασης (δημόσιο ή ιδιωτικό).

Ο χρήστης μπορεί να τροποποιεί τα μεταδεδομένα ενός μοντέλου, να το επανασυνδέει με νεότερη εκπαίδευση (επανεκπαίδευση), καθώς και να το διαγράφει, υπό την προϋπόθεση ότι είναι ο κάτοχός του. Η αναζήτηση μοντέλων υποστηρίζεται τόσο σε απλή όσο και σε σύνθετη μορφή, με φίλτρα που αφορούν μεταδεδομένα, κατηγορίες και χρονικά διαστήματα. Τα ιδιωτικά μοντέλα είναι προσβάσιμα αποκλειστικά από τον κάτοχό τους, ενώ τα δημόσια μπορούν να χρησιμοποιούνται και από άλλους χρήστες. Τέλος ο χρήστης μπορεί να διαμοιραστεί ένα μοντέλο μηχανικής μάθησης με άλλους χρήστες της επιλογής του ή με κάποιο group-ομάδας που ανήκει εφόσον έχει τα σχετικά δικαιώματα.

4.5.3.4 Εκτέλεση Μοντέλων Μηχανικής Μάθησης

Η εκτέλεση μοντέλων μηχανικής μάθησης υλοποιείται σύμφωνα με τις προδιαγραφές, επιτρέποντας την εφαρμογή ενός εκπαιδευμένου μοντέλου σε νέο σύνολο δεδομένων με

συμβατή δομή χαρακτηριστικών. Η εκτέλεση μπορεί να πραγματοποιείται είτε από τον κάτοχο του μοντέλου είτε από άλλους χρήστες, εφόσον το μοντέλο είναι δημόσιο ή έχει διαμοιραστεί σε αυτούς.

Η διαδικασία εκτέλεσης είναι ασύγχρονη και παράγει αποτελέσματα που μπορούν να προβληθούν και να εκφορτωθούν από τον χρήστη. Παρέχονται λειτουργίες επανεκτέλεσης, αναζήτησης εκτελέσεων με βάση χρονικά κριτήρια, εμφάνισης μεταδεδομένων και διαγραφής εκτελέσεων μαζί με τα συνοδευτικά αρχεία, καλύπτοντας πλήρως τις σχετικές απαιτήσεις.

4.5.3.5 Διαχείριση Προσαρμοσμένων Αλγορίθμων Μηχανικής Μάθησης

Η εφαρμογή υποστηρίζει τη διαχείριση προσαρμοσμένων αλγορίθμων μηχανικής μάθησης που παρέχονται από τους χρήστες, ικανοποιώντας μία από τις βασικότερες απαιτήσεις του συστήματος. Οι αλγόριθμοι εισάγονται ως δοχειοποιημένες εφαρμογές (Docker images), είτε μέσω μεταφόρτωσης είτε μέσω αναφοράς σε αποθετήριο (εικόνων δοχείων).

Κατά την εκτέλεση, τα δεδομένα εισόδου και οι παράμετροι διατίθενται σε προκαθορισμένο σημείο προσάρτησης (/data), ενώ τα αποτελέσματα παράγονται σε αντίστοιχο σημείο εξόδου (/model). Η χρήση δοχείων εξασφαλίζει απομόνωση, ασφάλεια και ανεξαρτησία από τη γλώσσα ή το περιβάλλον υλοποίησης του αλγορίθμου. Υποστηρίζονται λειτουργίες τροποποίησης, διαγραφής, εμφάνισης και αναζήτησης αλγορίθμων, με σεβασμό στα δικαιώματα πρόσβασης.

4.5.3.6 Επιπλέον Λειτουργίες Εφαρμογής

Πέραν των βασικών λειτουργιών που προδιαγράφηκαν κατά τη φάση σχεδίασης, κατά την πορεία της υλοποίησης αναπτύχθηκαν επιπρόσθετες λειτουργίες που κρίθηκαν απαραίτητες για την υποστήριξη συνεργατικών σεναρίων χρήσης.

Η πρώτη αφορά τη δημιουργία ομάδων χρηστών (groups). Κάθε ομάδα δημιουργείται από έναν χρήστη, ο οποίος αναλαμβάνει τον ρόλο του επικεφαλής. Σκοπός των ομάδων είναι η διευκόλυνση της συνεργασίας μεταξύ χρηστών που εργάζονται σε κοινά projects μηχανικής μάθησης, επιτρέποντας τον οργανωμένο διαμοιρασμό πόρων εντός ενός κοινού πλαισίου εργασίας.

Η δεύτερη αφορά τον διαμοιρασμό πόρων: συνόλων δεδομένων (datasets), μοντέλων, αλλά και ολόκληρου του pipeline εκπαίδευσης. Η δυνατότητα αυτή υλοποιήθηκε καθώς σε πραγματικά περιβάλλοντα μηχανικής μάθησης, η αναπαραγωγιμότητα και η ανταλλαγή πειραμάτων μεταξύ μελών μιας ομάδας αποτελούν κρίσιμες ανάγκες. Ο διαμοιρασμός συνόλων δεδομένων και μοντέλων είναι απλός — παραχωρείται πρόσβαση στον αντίστοιχο πόρο. Για τον διαμοιρασμό ενός pipeline εκπαίδευσης, η διαδικασία είναι πιο σύνθετη: το σύστημα αντιγράφει στη βάση δεδομένων όλα τα μεταδεδομένα κάθε βήματος που ακολουθήθηκε (διαμόρφωση συνόλου δεδομένων, παραμέτρους αλγορίθμου, εκπαίδευση) από τον αποστολέα στον παραλήπτη, και αντιγράφει το παραγόμενο μοντέλο στο Object Storage. Σε κάθε περίπτωση, η ιστορικότητα των διαμοιρασμών διατηρείται σε αντίστοιχους πίνακες της βάσης δεδομένων.

4.5.4 Ικανοποίηση Μη Λειτουργικών Απαιτήσεων

Η εφαρμογή ικανοποιεί σε μεγάλο βαθμό και τις μη λειτουργικές απαιτήσεις που τέθηκαν κατά τον σχεδιασμό. Η αρχιτεκτονική διαχωρισμού frontend και backend, σε συνδυασμό με την υλοποίηση RESTful API επιπέδου ωριμότητας 2, εξασφαλίζει καθαρό διαχωρισμό ευθυνών, επεκτασιμότητα και διαλειτουργικότητα.

Η χρήση δοχειοποίησης επιτρέπει την φορητότητα της εφαρμογής, καθώς δοχειοποιημένες εφαρμογές μπορούν να τρέξουν σε οποιοδήποτε σύστημα. Ακόμα η δοχειοποίηση επιτρέπει την εκτέλεση της εφαρμογής σε περιβάλλοντα Kubernetes, παρέχοντας τη δυνατότητα κλιμάκωσης και αξιόπιστης διαχείρισης υπολογιστικών φορτίων, καθώς δημιουργείται ένα νέο Kubernetes Job για κάθε εκπαίδευση. Η εφαρμογή υποστηρίζει ταυτόχρονους χρήστες, ενώ οι χρονοβόρες διεργασίες (εκπαίδευση και εκτέλεση μοντέλων) υλοποιούνται ασύγχρονα, ώστε να διασφαλίζεται η απόκριση του συστήματος.

Ιδιαίτερη έμφαση δόθηκε στην ασφάλεια, καθώς αποτελεί κρίσιμη πτυχή κάθε διαδικτυακής εφαρμογής, ιδιαίτερα όταν αυτή διαχειρίζεται ευαίσθητα δεδομένα χρηστών, σύνολα δεδομένων και εκπαιδευμένα μοντέλα μηχανικής μάθησης.

Συγκεκριμένα εφαρμόστηκαν οι εξής μηχανισμοί/τεχνικές ασφάλειας:

- **Κατακερματισμός κωδικών πρόσβασης**

Οι κωδικοί πρόσβασης των χρηστών αποθηκεύονται αποκλειστικά σε κατακερματισμένη μορφή μέσω του αλγορίθμου Argon2id, ο οποίος αποτελεί τον νικητή του διαγωνισμού Password Hashing Competition (2015) και συνιστάται από τον οργανισμό

OWASP. Η παραμετροποίηση που χρησιμοποιήθηκε (salt 16 bytes, hash 32 bytes, memory cost 4096 KB, iterations 3) καθιστά τον αλγόριθμο σκόπιμα αργό και μνημοβόρο, αναχαιτίζοντας αποτελεσματικά επιθέσεις ωμής βίας (brute force) και επιθέσεις λεξικού (dictionary attacks), καθώς κάθε δοκιμή κωδικού απαιτεί σημαντικούς υπολογιστικούς πόρους. Παράλληλα, η χρήση μοναδικού τυχαίου salt ανά κωδικό εξουδετερώνει τις επιθέσεις πινάκων αντιστροφής (rainbow table attacks), αφού ακόμα και πανομοιότυποι κωδικοί παράγουν διαφορετικά hashes. Επιπροσθέτως, εφαρμόζονται κανόνες πολυπλοκότητας κωδικού τόσο στην πλευρά του πελάτη όσο και του εξυπηρετητή, απαιτώντας τουλάχιστον 8 χαρακτήρες με συνδυασμό κεφαλαίων, πεζών, ψηφίων και ειδικών χαρακτήρων, αυξάνοντας εκθετικά τον χώρο αναζήτησης που πρέπει να εξαντλήσει ένας επιτιθέμενος.

- **Αυθεντικοποίηση και διαχείριση συνεδριών**

Η αυθεντικοποίηση βασίζεται σε JSON Web Tokens (JWT), τα οποία υπογράφονται με ασύμμετρη κρυπτογραφία RSA αντί για κοινό μυστικό κλειδί (shared secret). Η χρήση ζεύγους ιδιωτικού/δημοσίου κλειδιού αναχαιτίζει επιθέσεις πλαστογράφησης token (token forgery), καθώς ένας επιτιθέμενος δεν δύναται να δημιουργήσει έγκυρο token χωρίς πρόσβαση στο ιδιωτικό κλειδί, ακόμα και αν γνωρίζει τη δομή του token. Κάθε token έχει χρόνο λήξης μίας ώρας, περιορίζοντας το χρονικό παράθυρο εκμετάλλευσης σε περίπτωση υποκλοπής. Επιπλέον, υλοποιήθηκε μηχανισμός ανάκλησης tokens (token revocation), ο οποίος σημαδεύει τα tokens ως ανακληθέντα (revoked) στη βάση δεδομένων κατά την αποσύνδεση του χρήστη. Ο JwtTokenFilter ελέγχει σε κάθε αίτημα τόσο την εγκυρότητα της υπογραφής όσο και την κατάσταση ανάκλησης, αναχαιτίζοντας επιθέσεις υποκλοπής συνεδρίας (session hijacking) και επαναχρησιμοποίησης token (token replay attacks).

- **Έλεγχος πρόσβασης**

Η εφαρμογή υλοποιεί πολυεπίπεδο έλεγχο πρόσβασης βασισμένο σε ρόλους (Role-Based Access Control), αξιοποιώντας τον μηχανισμό @PreAuthorize του Spring Security σε επίπεδο μεθόδου. Κάθε endpoint ελέγχει τον ρόλο του χρήστη (USER ή ADMIN), ενώ σε επίπεδο υπηρεσιών (service layer) πραγματοποιούνται πρόσθετοι έλεγχοι ιδιοκτησίας, διασφαλίζοντας ότι ένας χρήστης μπορεί να τροποποιήσει μόνο τους δικούς του πόρους. Μέσω αυτού του μηχανισμού αναχαιτίζονται επιθέσεις κλιμάκωσης δικαιωμάτων (privilege escalation) και μη εξουσιοδοτημένης πρόσβασης σε πόρους (Broken Access Control), οι οποίες κατατάσσονται στην πρώτη θέση του OWASP Top 10.

- **Προστασία από SQL Injection**

Η πρόσβαση στη βάση δεδομένων πραγματοποιείται αποκλειστικά μέσω του JPA/Hibernate, το οποίο χρησιμοποιεί παραμετροποιημένα ερωτήματα (parameterized queries) σε όλα τα repositories. Τα δεδομένα εισόδου των χρηστών δεν ενσωματώνονται ποτέ ως συμβολοσειρές σε εντολές SQL, αλλά αντιμετωπίζονται ως παράμετροι μέσω του μηχανισμού @Param, αναχαιτίζοντας πλήρως τις επιθέσεις SQL Injection.

- **Επικύρωση δεδομένων εισόδου**

Η επικύρωση δεδομένων υλοποιείται σε δύο επίπεδα. Στην πλευρά του πελάτη, εφαρμόζεται επικύρωση φορμών μέσω κανονικών εκφράσεων (regular expressions) και ελέγχων υποχρεωτικών πεδίων πριν από την αποστολή κάθε αιτήματος. Στην πλευρά του εξυπηρετητή, αξιοποιούνται εκτενώς τα Jakarta Validation annotations (@NotBlank, @Size, @Pattern, @Email, @Min, @Positive) στα αντικείμενα μεταφοράς δεδομένων (DTOs), ενώ υλοποιήθηκαν τέσσερις προσαρμοσμένοι validators: ο ValidFileExtension για τον περιορισμό τύπων αρχείων σε .csv και .arff, ο ValidTarball για τον έλεγχο περιεχομένων Docker images, ο ValidImageSource για τη διασφάλιση αποκλειστικής παροχής πηγής image, και ο ValidBasicCharacteristicsColumns για την επικύρωση δεικτών στηλών. Μέσω αυτών αναχαιτίζονται επιθέσεις παραποίησης παραμέτρων (parameter tampering) και μεταφόρτωσης κακόβουλων αρχείων (malicious file upload), ενώ η χρήση DTOs αντί για απευθείας σύνδεση με οντότητες αποτρέπει επιθέσεις mass assignment.

4.5.5 Περιπτώσεις Μερικής Ικανοποίησης και Αιτιολόγηση

Οι περιπτώσεις μερικής ικανοποίησης απαιτήσεων της παρούσας εργασίας δεν οφείλονται σε τεχνικούς περιορισμούς της επιλεγμένης τεχνολογικής στοίβας, αλλά αποτελούν συνειδητές σχεδιαστικές επιλογές με στόχο τη διατήρηση της αρχιτεκτονικής απλότητας, της επεκτασιμότητας και της σταθερότητας του συστήματος.

4.5.5.1 Παρακολούθηση Προόδου Εκπαιδεύσεων και Εκτελέσεων

Η παρακολούθηση της κατάστασης εκπαίδευσης και εκτέλεσης μοντέλων υλοποιείται μέσω μηχανισμού περιοδικής ανάκτησης κατάστασης (polling), αντί για μηχανισμό push-based ενημέρωσης σε πραγματικό χρόνο (π.χ. WebSockets ή Server-Sent Events). Η επιλογή αυτή μειώνει την πολυπλοκότητα της αρχιτεκτονικής και αποφεύγει τη διατήρηση μόνιμων αμφίδρομων συνδέσεων, οι οποίες θα αύξαναν το διαχειριστικό και λειτουργικό κόστος.

Δεδομένου ότι οι διαδικασίες εκπαίδευσης ενδέχεται να διαρκούν αρκετά λεπτά, η χρήση polling θεωρήθηκε επαρκής για τις ανάγκες της εφαρμογής.

4.5.5.2 Ωριμότητα REST και Ασφάλεια

Η εφαρμογή δεν ικανοποιεί πλήρως το επίπεδο ωριμότητας REST Level 2 (HTTP Verbs), καθώς οι αποκρίσεις του API δεν περιλαμβάνουν δυναμικούς υπερσυνδέσμους πλοήγησης. Η υλοποίηση περιορίζεται σε REST Level 2, το οποίο θεωρήθηκε επαρκές για τις λειτουργικές απαιτήσεις της πλατφόρμας.

Σε επίπεδο ασφάλειας, έχουν υλοποιηθεί βασικοί μηχανισμοί αυθεντικοποίησης και εξουσιοδότησης μέσω JWT, ωστόσο ορισμένες κατηγορίες επιθέσεων δεν καλύπτονται πλήρως στην τρέχουσα έκδοση:

- Δεν έχει ενεργοποιηθεί κρυπτογράφηση TLS στο επίπεδο της εφαρμογής. Η ασφαλής επικοινωνία προβλέπεται να παρέχεται σε περιβάλλον παραγωγής μέσω reverse proxy ή Ingress Controller.
- Η προστασία CSRF είναι απενεργοποιημένη, γεγονός που αποτελεί συνήθη πρακτική σε stateless REST APIs με χρήση JWT σε HTTP headers αντί για cookies.
- Δεν έχουν υλοποιηθεί ρητοί μηχανισμοί προστασίας έναντι επιθέσεων XSS (π.χ. Content Security Policy, security headers) και SSRF (επικύρωση εξωτερικών URLs). Τα σημεία αυτά αποτελούν αντικείμενο μελλοντικής βελτίωσης.

4.5.5.3 Ταυτόχρονοι Χρήστες (Concurrency)

Η απαίτηση υποστήριξης τουλάχιστον 20 ταυτόχρονων χρηστών ικανοποιείται μερικώς. Αν και δεν έχει πραγματοποιηθεί ρητή παραμετροποίηση ή stress testing με συγκεκριμένα όρια ταυτόχρονων αιτημάτων, η αρχιτεκτονική του συστήματος — βασισμένη σε απομονωμένες ασύγχρονες εργασίες και διαχωρισμό επιπέδων — επιτρέπει θεωρητικά την κλιμάκωση σε τέτοιο επίπεδο χρήσης. Η πλήρης επιβεβαίωση της απαίτησης προϋποθέτει μελλοντική εκτέλεση δοκιμών φόρτου και βελτιστοποίηση παραμέτρων εκτέλεσης.

4.5.5.4 Responsive Διεπαφή Χρήστη

Η απαίτηση για πλήρως responsive διεπαφή ικανοποιείται μερικώς. Έχουν υλοποιηθεί βασικοί μηχανισμοί προσαρμογής όπως:

- media queries (ενδεικτικά εύρη 480px–1024px),
- χρήση flexbox και CSS grid,
- χρήση της συνάρτησης clamp(),
- υλοποίηση mobile navigation (hamburger menu).

Ωστόσο, ορισμένα components απαιτούν περαιτέρω σχεδιαστική βελτιστοποίηση ώστε να παρέχουν ισοδύναμη εμπειρία χρήστη σε μικρότερες συσκευές, ιδιαίτερα σε σύνθετες λειτουργικές οθόνες.

4.5.5.5 Συνολική Αιτιολόγηση

Οι παραπάνω περιπτώσεις μερικής ικανοποίησης δεν υποδηλώνουν δομική αδυναμία του συστήματος, αλλά αντανακλούν την προτεραιοποίηση της αρχιτεκτονικής καθαρότητας, της απομόνωσης υπολογιστικών εργασιών και της υποστήριξης παραγωγικών σεναρίων μηχανικής μάθησης. Η έμφαση δόθηκε στη διαμόρφωση μιας σταθερής και επεκτάσιμης βάσης, ικανής να υποστηρίξει απαιτητικά υπολογιστικά φορτία και ετερογενή περιβάλλοντα εκτέλεσης.

Οι μη πλήρως υλοποιημένες απαιτήσεις μπορούν να ενσωματωθούν σε μελλοντικές εκδόσεις χωρίς αναδιάρθρωση της βασικής αρχιτεκτονικής, γεγονός που επιβεβαιώνει την ευελιξία του σχεδιασμού.

4.5.6 Συνολική Αποτίμηση

Συνολικά, η ανάλυση καταδεικνύει ότι η εφαρμογή ανταποκρίνεται με επιτυχία στους βασικούς στόχους που τέθηκαν κατά τον σχεδιασμό της. Η πλήρης κάλυψη των κρίσιμων λειτουργικών και μη λειτουργικών απαιτήσεων, σε συνδυασμό με τις τεκμηριωμένες περιπτώσεις μερικής υλοποίησης, αναδεικνύει μια συνεκτική και ώριμη αρχιτεκτονική προσέγγιση.

Η εφαρμογή μπορεί να θεωρηθεί μια ολοκληρωμένη και αξιόπιστη πλατφόρμα μηχανικής μάθησης σε περιβάλλον υπολογιστικού νέφους, με σαφείς δυνατότητες περαιτέρω επέκτασης και βελτίωσης στο πλαίσιο μελλοντικής εργασίας.

5 Εγκατάσταση & Επίδειξη της Εφαρμογής

Το παρόν κεφάλαιο επικεντρώνεται στην πρακτική επικύρωση της εφαρμογής ThesisApp μέσω οργανωμένης επίδειξης και αξιολόγησης της λειτουργίας της. Σκοπός του κεφαλαίου είναι, αφενός, η τεκμηρίωση της διαδικασίας εγκατάστασης και εκτέλεσης της εφαρμογής σε περιβάλλον που προσομοιώνει συνθήκες παραγωγής και, αφετέρου, η παρουσίαση αντιπροσωπευτικών σεναρίων χρήσης που καλύπτουν το σύνολο των βασικών λειτουργικών δυνατοτήτων του συστήματος.

Η επίδειξη πραγματοποιείται σε περιβάλλον Kubernetes, αξιοποιώντας το Minikube ως τοπικό cluster, γεγονός που επιτρέπει την αξιολόγηση της εφαρμογής όχι μόνο ως προς τη λειτουργική της ορθότητα, αλλά και ως προς τη συμβατότητά της με cloud-native αρχιτεκτονικές. Παράλληλα, εξετάζεται η συμπεριφορά του συστήματος κατά την εκτέλεση χρονοβόρων εργασιών, η διαχείριση πόρων, καθώς και η δυνατότητα παρακολούθησης και ακύρωσης εργασιών σε πραγματικό χρόνο.

5.1 Οδηγίες Εγκατάστασης και Εκτέλεσης της Εφαρμογής

5.1.1 Απαιτούμενο Λογισμικό και Περιβάλλον Εκτέλεσης

Η εφαρμογή ThesisApp σχεδιάστηκε για εκτέλεση σε περιβάλλοντα που υποστηρίζουν τεχνολογίες δοχείων (Docker) και ενορχήστρωσης μέσω Kubernetes. Για σκοπούς ανάπτυξης και επίδειξης, χρησιμοποιείται το Minikube, το οποίο παρέχει ένα τοπικό Kubernetes cluster με συμπεριφορά παρόμοια με περιβάλλον παραγωγής. Τα μόνα απαραίτητα εργαλεία για την εγκατάσταση και εκτέλεση της εφαρμογής είναι:

- Docker: https://docs.docker.com/get-docker/
- Minikube: https://minikube.sigs.k8s.io/docs/start/
- kubectl: https://kubernetes.io/docs/tasks/tools/

Οι παραπάνω σύνδεσμοι παρέχουν αναλυτικές και ενημερωμένες οδηγίες εγκατάστασης για διαφορετικά λειτουργικά συστήματα (Windows, Linux, macOS). Η παρούσα εργασία δεν αναπαράγει τα βήματα εγκατάστασης, αλλά βασίζεται στις επίσημες οδηγίες των εργαλείων. Η μεταγλώττιση του κώδικα (Java, Node.js) πραγματοποιείται εξ ολοκλήρου εντός των Docker εικόνων μέσω multi-stage builds, εξαλείφοντας την ανάγκη τοπικής εγκατάστασης εργαλείων ανάπτυξης όπως Maven ή Node.js.

Σε περιβάλλοντα Windows, απαιτείται η ενεργοποίηση του WSL2 (Windows Subsystem for Linux 2), το οποίο επιτρέπει τη χρήση εργαλείων Linux και την ομαλή συνεργασία με Docker και Kubernetes. Η προϋπόθεση αυτή δεν ισχύει για συστήματα Linux ή macOS, στα οποία τα εργαλεία εκτελούνται εγγενώς. Η τοπική εκτέλεση μέσω Minikube απαιτεί επαρκείς υπολογιστικούς πόρους, καθώς εκτελούνται πολλαπλά containers ενώ η εκπαίδευση μοντέλων μηχανικής μάθησης μπορεί να είναι απαιτητική σε μνήμη και επεξεργαστική ισχύ. Οι ελάχιστες προτεινόμενες απαιτήσεις για ομαλή λειτουργία είναι:

- *Επεξεργαστής (CPU)*: τουλάχιστον 4 λογικοί πυρήνες
- *Μνήμη RAM*: τουλάχιστον 8 GB (προτείνεται 16 GB για εκπαίδευση μεγαλύτερων μοντέλων)
- *Αποθηκευτικός χώρος*: τουλάχιστον 20 GB ελεύθερου χώρου για Docker images, datasets και παραγόμενα μοντέλα
- *WSL2 allocated memory(Μόνο για Windows)*: τουλάχιστον 6 GB διαθέσιμα στο Linux subsystem

Σε συστήματα με μικρότερους πόρους είναι πιθανή η υποβάθμιση απόδοσης ή η αποτυχία εκκίνησης του Kubernetes cluster.

5.1.2 Διαδικασία Εγκατάστασης

Με την προϋπόθεση ότι τα απαραίτητα εργαλεία (Docker, kubectl, Minikube) έχουν εγκατασταθεί σύμφωνα με τις οδηγίες της προηγούμενης ενότητας, η εγκατάσταση της εφαρμογής πραγματοποιείται μέσω πέντε εντολών. Αρχικά, εκκινείται το τοπικό Kubernetes cluster: minikube start.

Στη συνέχεια, σε ξεχωριστό τερματικό, ενεργοποιείται ο μηχανισμός προώθησης δικτύου (port-forwarding), ώστε οι υπηρεσίες του cluster να είναι προσβάσιμες τοπικά. Σε περιβάλλον Windows με WSL2, η εντολή αυτή είναι: ./kubernetes/start-tunnel-wsl2.sh. Σε συστήματα Linux ή macOS, η πρόσβαση στις υπηρεσίες πραγματοποιείται μέσω minikube tunnel ή απευθείας μέσω των NodePort endpoints.

Το σενάριο αναμένει την ετοιμότητα του Ingress controller και προωθεί τη θύρα 80 του τοπικού συστήματος στον Ingress controller του cluster μέσω kubectl port-forward. Για την αντιστοίχιση του domain name thesisapp.local απαιτείται η εφάπαξ προσθήκη μίας εγγραφής στο αρχείο hosts του συστήματος. Σε περιβάλλοντα Windows, το αρχείο βρίσκεται στη διαδρομή C:\Windows\System32\drivers\etc\hosts, ενώ σε συστήματα Linux και macOS στη

διαδρομή /etc/hosts. Η απαιτούμενη εγγραφή υποδεικνύεται αυτόματα κατά την εκτέλεση του σεναρίου.

Στο αρχικό τερματικό, κλωνοποιείται το αποθετήριο κώδικα και εκτελείται το αυτοματοποιημένο σενάριο εγκατάστασης:

```
git clone https://github.com/NikolasSpirouGH/ThesisApp.git
cd ThesisApp
./kubernetes/deploy-to-minikube.sh
```

Το σενάριο εγκατάστασης αναλαμβάνει αυτόματα: (α) την ενεργοποίηση του Ingress addon, (β) την κατασκευή των εικόνων δοχείων για το backend, το frontend και τον Weka Runner μέσω multi-stage Docker builds, (γ) τη φόρτωση των εικόνων στο εσωτερικό αποθετήριο του Minikube, (δ) τη δημιουργία των Kubernetes Secrets και ConfigMaps, και (ε) την εφαρμογή όλων των Kubernetes manifests (Deployments, StatefulSets, Services, Ingress). Οι εικόνες τρίτων (PostgreSQL, MinIO, MailHog) ανακτώνται αυτόματα από δημόσια αποθετήρια. Με τον τρόπο αυτό, η εγκατάσταση είναι πλήρως αυτοματοποιημένη, επαναλήψιμη και συνεπής.

5.1.3 Εκκίνηση και Τερματισμός της Εφαρμογής

Μετά την ολοκλήρωση της αρχικής εγκατάστασης, η εφαρμογή είναι άμεσα προσβάσιμη, καθώς η εκκίνηση του Minikube και η διαμόρφωση του μηχανισμού προώθησης δικτύου έχουν ήδη πραγματοποιηθεί κατά τη διαδικασία εγκατάστασης.

Η πρόσβαση στην εφαρμογή πραγματοποιείται μέσω του domain name thesisapp.local και τα διαθέσιμα endpoints είναι τα ακόλουθα:

- Frontend (Διεπαφή χρήστη): http://thesisapp.local
- Backend API: http://thesisapp.local/api
- Τεκμηρίωση API (Swagger UI): http://thesisapp.local/api/swagger-ui/index.html
- MailHog (Διεπαφή ελέγχου email): http://mailhog.thesisapp.local
- MinIO Console (Διαχείριση αποθηκευτικού χώρου): http://minio.thesisapp.local

Οι λογαριασμοί δημιουργούνται αυτόματα μέσω Flyway database migration κατά την πρώτη εκκίνηση της εφαρμογής και οι κωδικοί αποθηκεύονται κρυπτογραφημένοι με τον αλγόριθμο Argon2id.

Ο τερματισμός της εφαρμογής μπορεί να πραγματοποιηθεί με δύο τρόπους:

- Παύση με διατήρηση δεδομένων: minikube stop — διατηρεί τα Persistent Volumes και τα δεδομένα της βάσης. Για επανεκκίνηση αρκεί η εκτέλεση minikube start ακολουθούμενη από το σενάριο προώθησης δικτύου (./kubernetes/start-tunnel-wsl2.sh σε περιβάλλον Windows/WSL2, ή minikube tunnel σε Linux/macOS).
- Πλήρης διαγραφή περιβάλλοντος: minikube delete — διαγράφει το cluster και όλα τα αποθηκευμένα δεδομένα, απαιτώντας εκ νέου εκτέλεση του σεναρίου εγκατάστασης.

5.2 Επίδειξη της Εφαρμογής

Για την επίδειξη των δυνατοτήτων της εφαρμογής επιλέχθηκαν εννέα αντιπροσωπευτικά σενάρια χρήσης, τα οποία καλύπτουν τις βασικές λειτουργίες του συστήματος. Τα σενάρια αυτά περιλαμβάνουν την εκπαίδευση μοντέλων με προκαθορισμένους αλγορίθμους, την εκπαίδευση με προσαρμοσμένους αλγορίθμους που παρέχονται από τον χρήστη, την εκτέλεση προβλέψεων σε νέα δεδομένα, ακύρωση εργασιών που βρίσκονται σε εξέλιξη, την διαχείριση χρηστών και κατηγοριών.

Η επιλογή των σεναρίων αποσκοπεί στην ανάδειξη τόσο της λειτουργικής πληρότητας της εφαρμογής όσο και της αρχιτεκτονικής της ευελιξίας, ιδίως ως προς τη διαχείριση ασύγχρονων και υπολογιστικά απαιτητικών εργασιών.

5.2.1 Σενάριο 1: Εκπαίδευση Μοντέλου με Προκαθορισμένο Αλγόριθμο (Weka)

Στο σενάριο αυτό παρουσιάζεται η διαδικασία εκπαίδευσης ενός μοντέλου ταξινόμησης χρησιμοποιώντας προκαθορισμένο αλγόριθμο της βιβλιοθήκης Weka.

Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος στο σύστημα και έχει ήδη μεταφορτώσει τουλάχιστον ένα σύνολο δεδομένων τύπου Training μέσω της σελίδας Datasets.

Βήματα:

1 Ο χρήστης επιλέγει το στοιχείο Trainings από το κεντρικό μενού πλοήγησης(δείτε Εικόνα 5.1) . Εμφανίζεται η σελίδα διαχείρισης εκπαιδεύσεων, η οποία περιλαμβάνει τη λίστα προηγούμενων εκπαιδεύσεων και το κουμπί εκκίνησης νέας εκπαίδευσης.

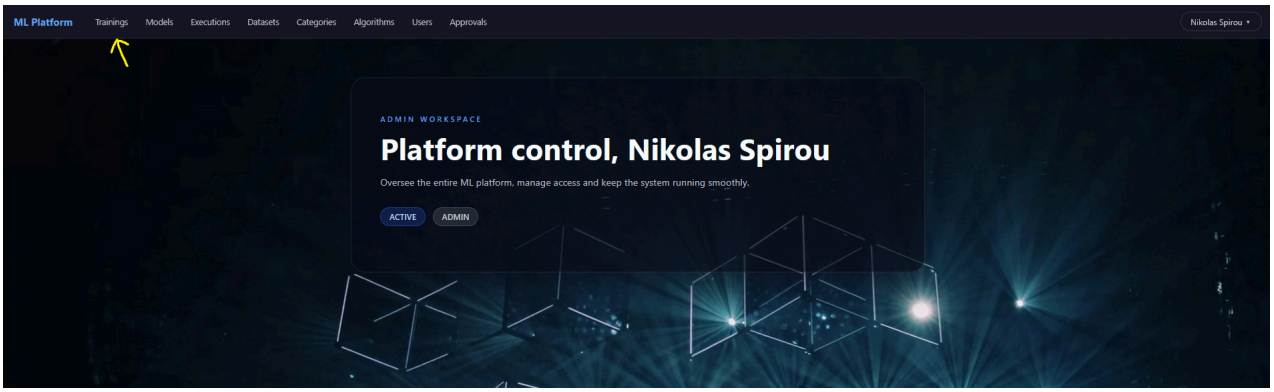
2 Ο χρήστης πατά το κουμπί Train (δείτε Εικόνα 5.2), οπότε εμφανίζεται ο διάλογος επιλογής τύπου εκπαίδευσης. Από τις διαθέσιμες επιλογές (Weka Training, Custom Training, Retrain), επιλέγει Weka Training και μεταβαίνει στη φόρμα εκπαίδευσης.

3 Στη φόρμα εκπαίδευσης, ο χρήστης επιλέγει ένα υπάρχον σύνολο δεδομένων από τη λίστα των ήδη μεταφορτωμένων συνόλων (ιδιωτικά ή δημόσια) και στη συνέχεια καθορίζει τον αλγόριθμο εκπαίδευσης καθώς και τις αντίστοιχες παραμέτρους του (δείτε Εικόνα 5.3).

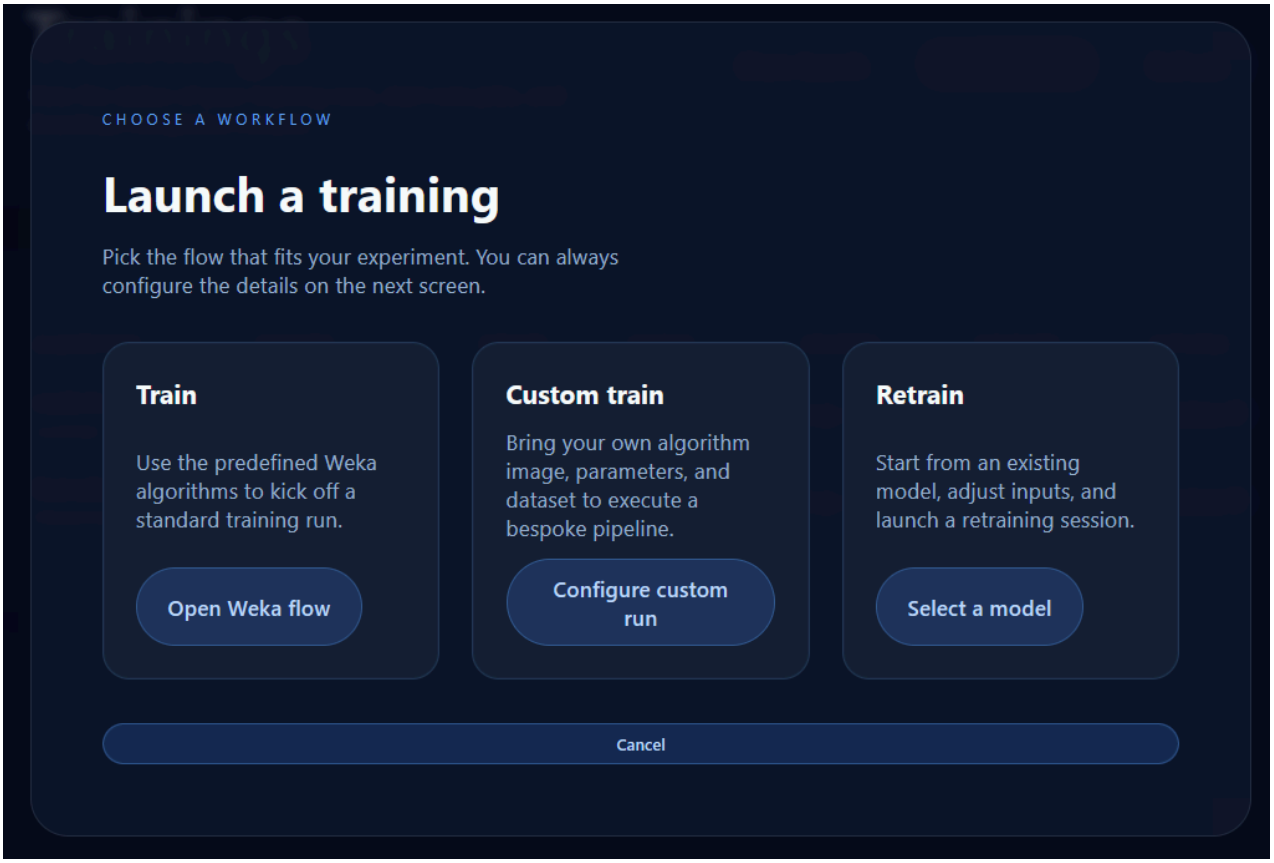
4 Ο χρήστης διαμορφώνει τον ρόλο των στηλών του συνόλου δεδομένων (δείτε Εικόνα 5.4), επιλέγοντας ποιες στήλες αποτελούν χαρακτηριστικά εισόδου (features) και ποια στήλη αποτελεί τη μεταβλητή-στόχο (target), και πατά το κουμπί Start Training για την εκκίνηση της εκπαίδευσης.

5 Με την εκκίνηση, η εφαρμογή δημιουργεί ένα Kubernetes Job, το οποίο εκτελεί τον αλγόριθμο σε απομονωμένο container (δείτε Εικόνα 5.5). Η εκτέλεση πραγματοποιείται ασύγχρονα, ενώ ο χρήστης μπορεί να παρακολουθεί την κατάσταση της εργασίας σε πραγματικό χρόνο μέσω του πάνελ ενεργών εργασιών (δείτε Εικόνα 5.7) που εμφανίζεται στη φόρμα. Η κατάσταση ενημερώνεται αυτόματα μέσω μηχανισμού polling κάθε 3 δευτερόλεπτα, χωρίς να απαιτείται ανανέωση της σελίδας.

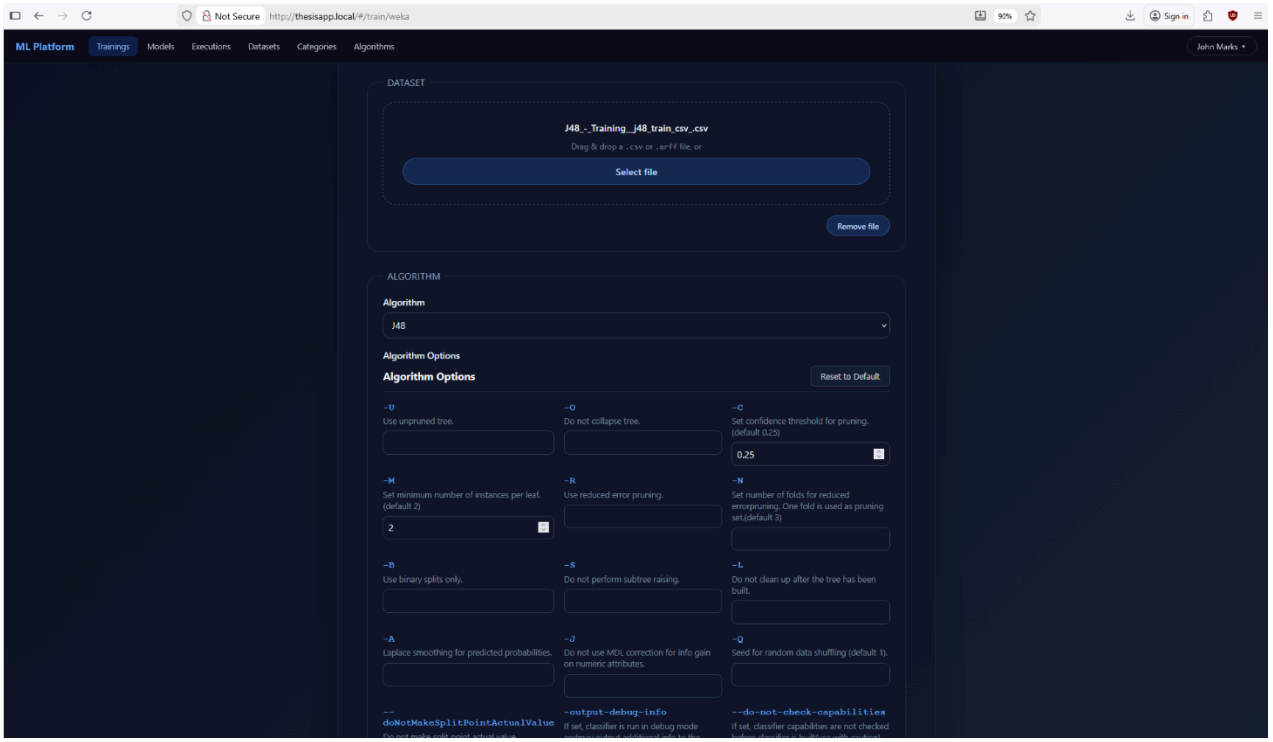
6 Με την ολοκλήρωση της εκπαίδευσης, το εκπαιδευμένο μοντέλο αποθηκεύεται στο σύστημα. Ο χρήστης μεταβαίνει στη σελίδα Trainings και επιλέγει View Results (δείτε Εικόνα 5.6) για την ολοκληρωμένη εκπαίδευση, οπότε εμφανίζονται οι μετρικές αξιολόγησης, όπως ακρίβεια (accuracy), ανάκληση (recall) και πίνακας σύγχυσης (confusion matrix), οι οποίες καθιστούν δυνατή την ποιοτική αποτίμηση της απόδοσής του.



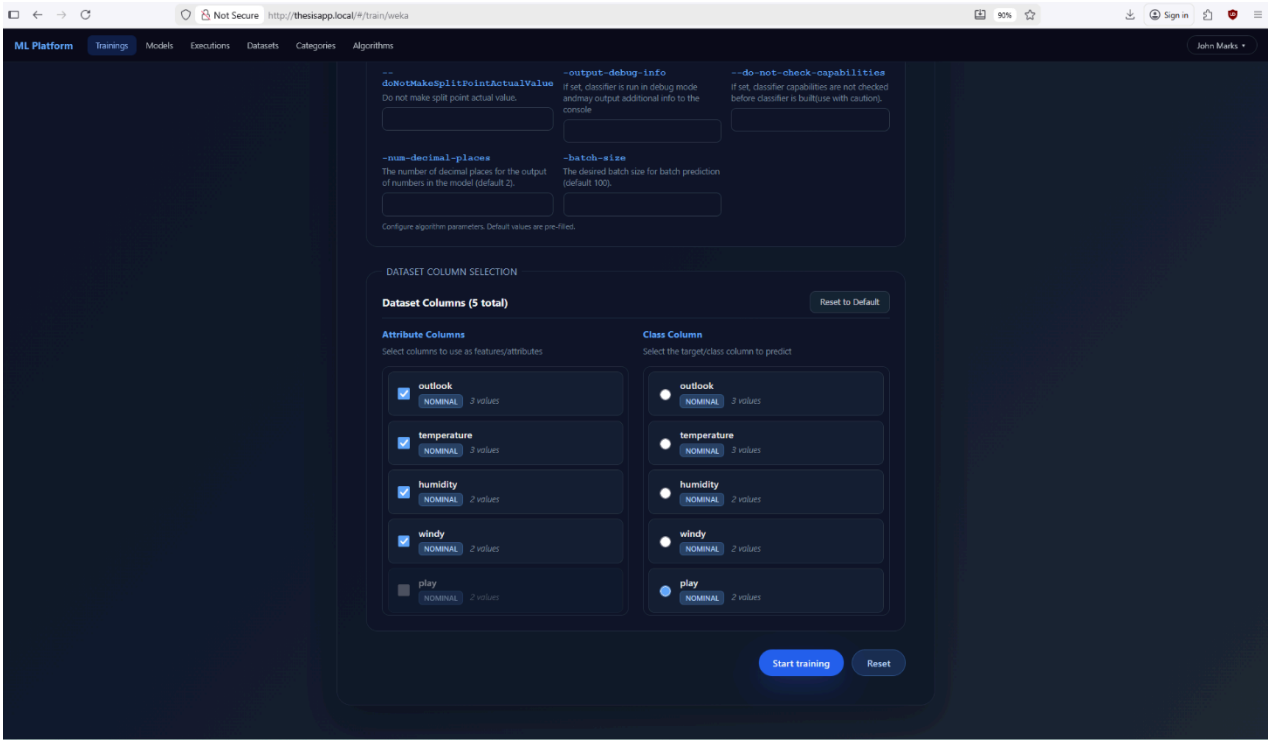
Εικόνα 5.1: Μενού Πλοήγησης - Επιλογή Εκπαιδεύσεων



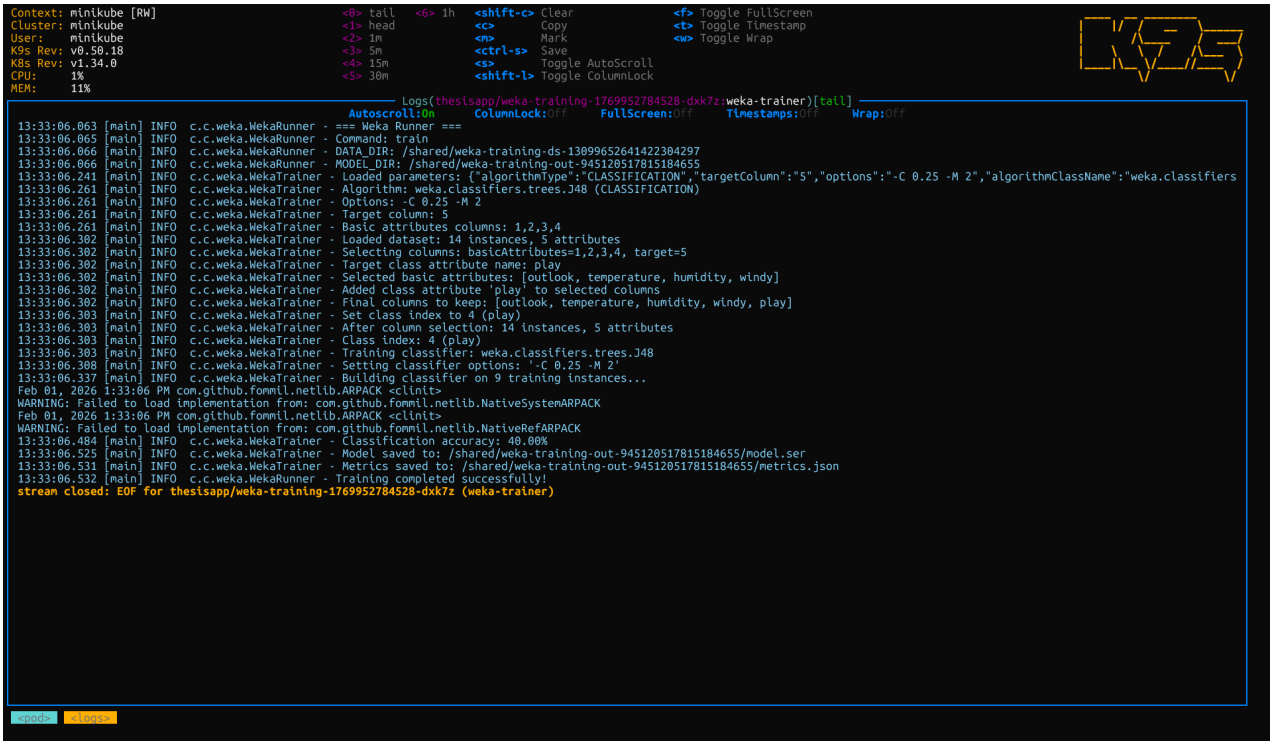
Εικόνα 5.2: Επιλογή Weka Εκπαίδευσης



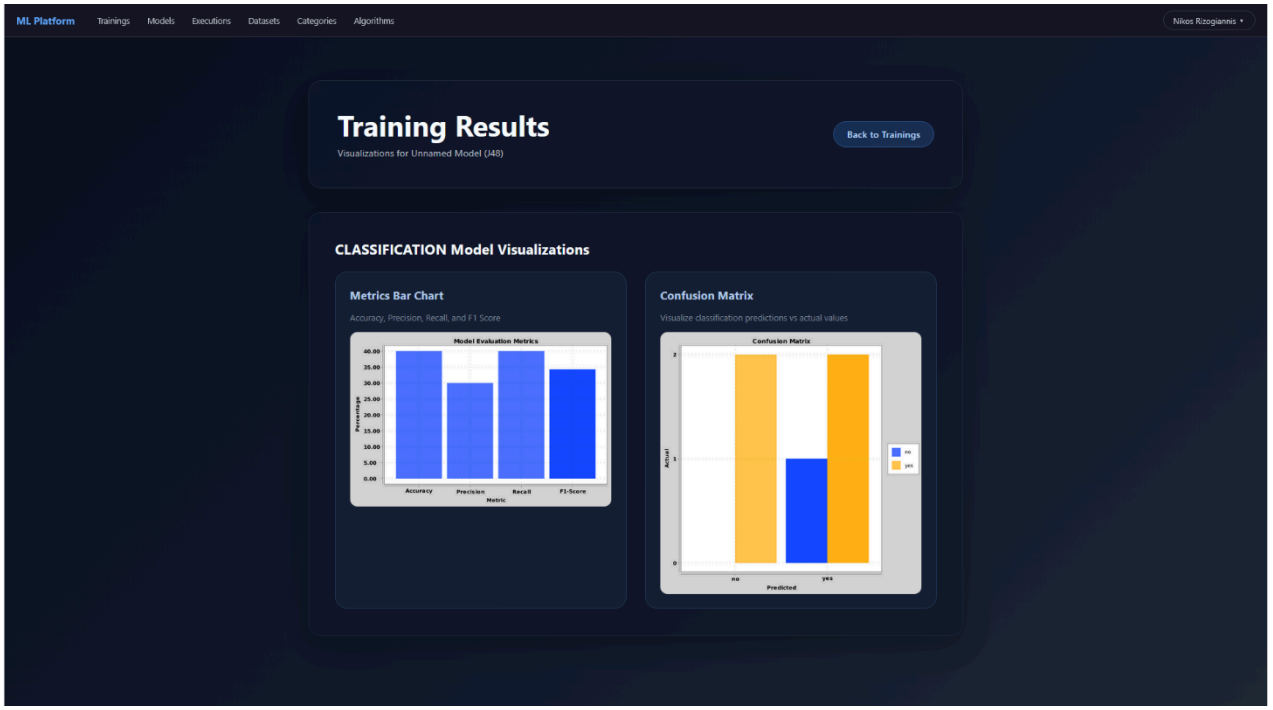
Εικόνα 5.3: Ενσωμάτωση Αλγορίθμου και Παραμέτρων



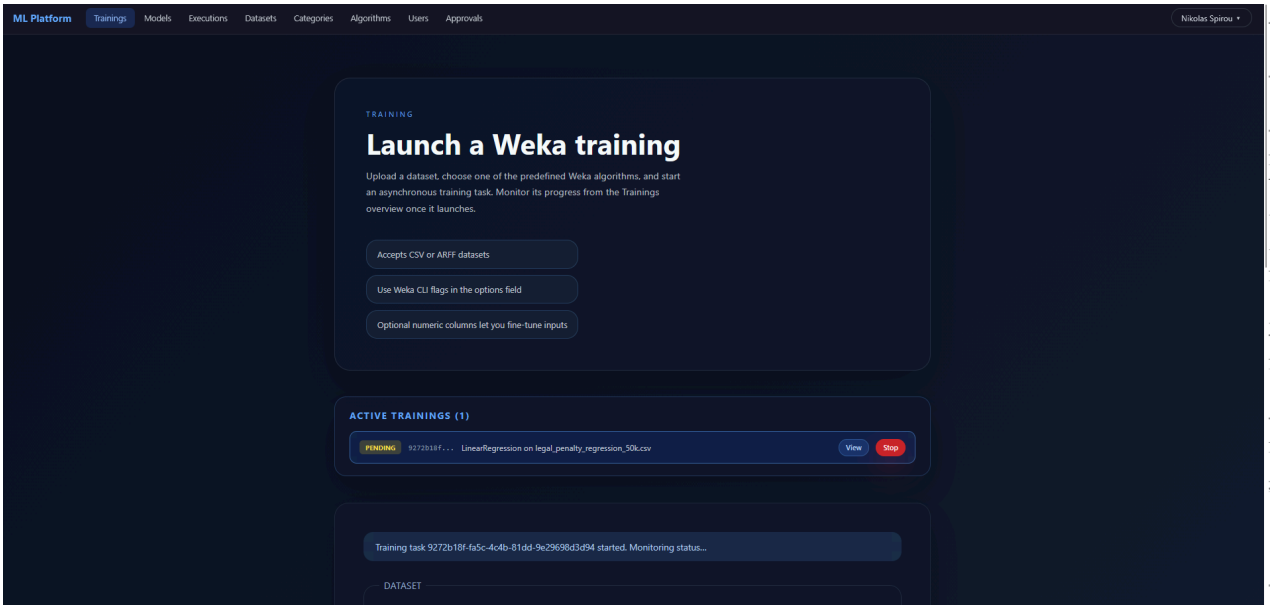
Εικόνα 5.4: Παραμετροποίηση Συνόλου Δεδομένων & Εκκίνηση Εκπαίδευσης



Εικόνα 5.5: Weka Training Kubernetes Pod (K9S)



Εικόνα 5.6: Διαγράμματα Μετρικών



Εικόνα 5.7: Πάνελ Ενεργών Εκπαιδεύσεων

5.2.2 Σενάριο 2: Δημιουργία Προσαρμοσμένου Αλγορίθμου

Η εφαρμογή επιτρέπει την ενσωμάτωση αλγορίθμων που υλοποιούνται από τον χρήστη σε οποιαδήποτε γλώσσα προγραμματισμού, μέσω της χρήσης Docker containers. Στο παρόν σενάριο, χρησιμοποιείται η Python με τη βιβλιοθήκη scikit-learn ως ενδεικτικό παράδειγμα, ωστόσο η ίδια διαδικασία εφαρμόζεται ανεξαρτήτως γλώσσας ή πλαισίου ανάπτυξης. Το σενάριο αυτό αποτελεί προαπαιτούμενο για την εκπαίδευση μοντέλου με προσαρμοσμένο αλγόριθμο.

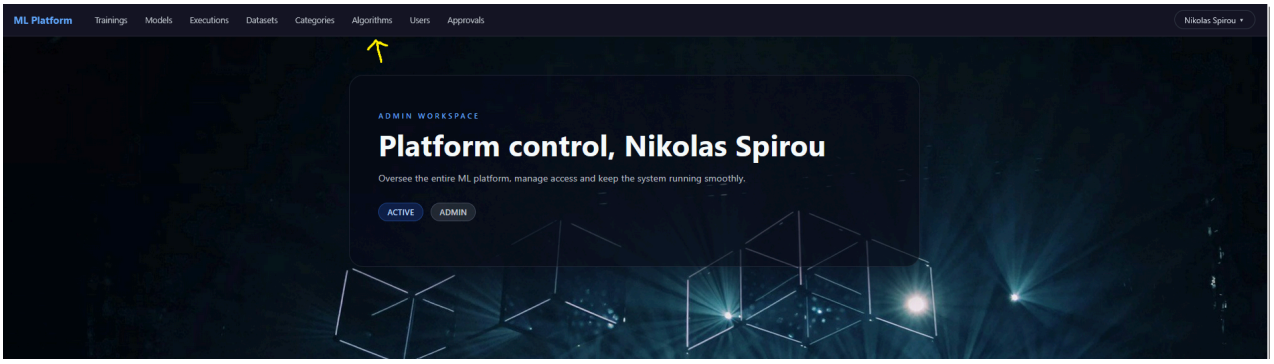
Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος στο σύστημα και διαθέτει Docker image του αλγορίθμου του, είτε σε μορφή αρχείου (.tar) είτε δημοσιευμένο σε αποθετήριο DockerHub.

Βήματα:

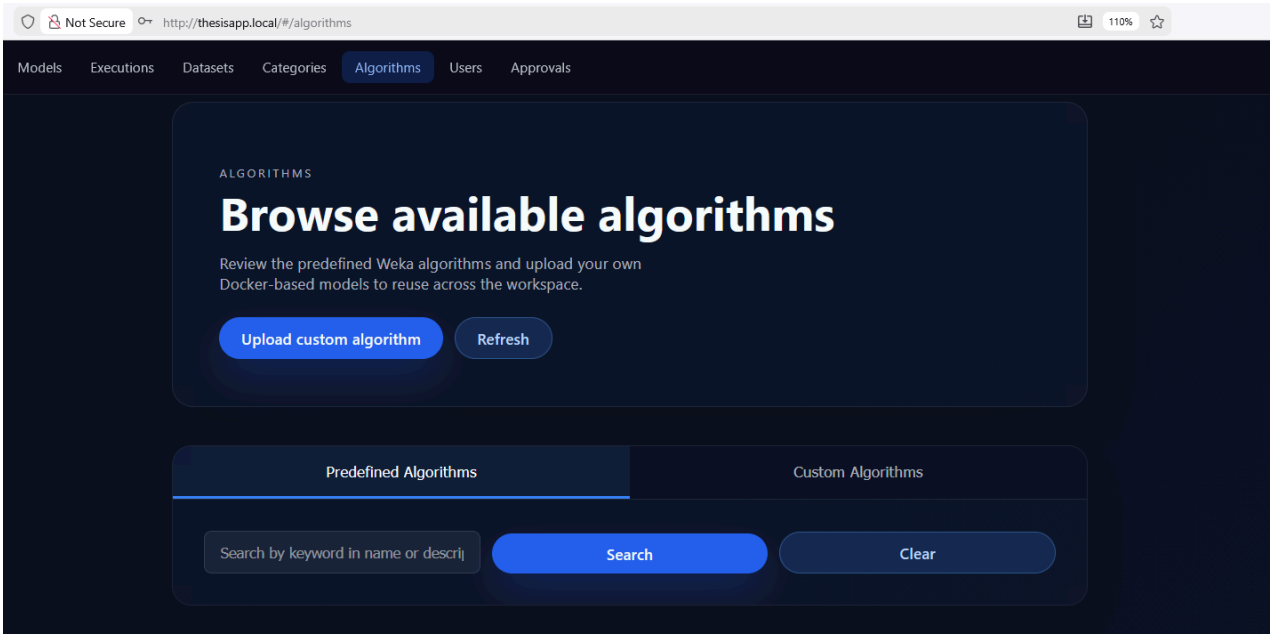
1. Ο χρήστης επιλέγει το στοιχείο Algorithms από το κεντρικό μενού πλοήγησης. Εμφανίζεται η σελίδα διαχείρισης αλγορίθμων (δείτε Εικόνα 5.8), η οποία περιλαμβάνει τη λίστα των υπαρχόντων αλγορίθμων και το κουμπί δημιουργίας νέου.
2. Ο χρήστης πατά το κουμπί Upload Algorithm και μεταβαίνει στη φόρμα δημιουργίας (δείτε Εικόνα 5.9).
3. Στη φόρμα, ο χρήστης συμπληρώνει τα απαραίτητα μεταδεδομένα του αλγορίθμου: όνομα, έκδοση και περιγραφή.

4. Ο χρήστης επιλέγει τον τρόπο παροχής του Docker image και του αρχείου με τις υπερπαραμέτρους (δείτε Εικόνα 5.10):
    - Μεταφόρτωση αρχείου: Επιλέγει αρχείο Docker image σε μορφή .tar, το οποίο αποθηκεύεται στο σύστημα αποθήκευσης αντικειμένων (MinIO).
    - Αποθετήριο DockerHub: Καθορίζει το όνομα της εικόνας (π.χ. username/algorithm:latest), επιτρέποντας την αυτόματη ανάκτησή της κατά τον χρόνο εκτέλεσης.
5. Με την υποβολή, ο αλγόριθμος καταχωρίζεται στο σύστημα και είναι πλέον διαθέσιμος για χρήση σε εκπαιδεύσεις μοντέλων (δείτε Εικόνα 5.11).

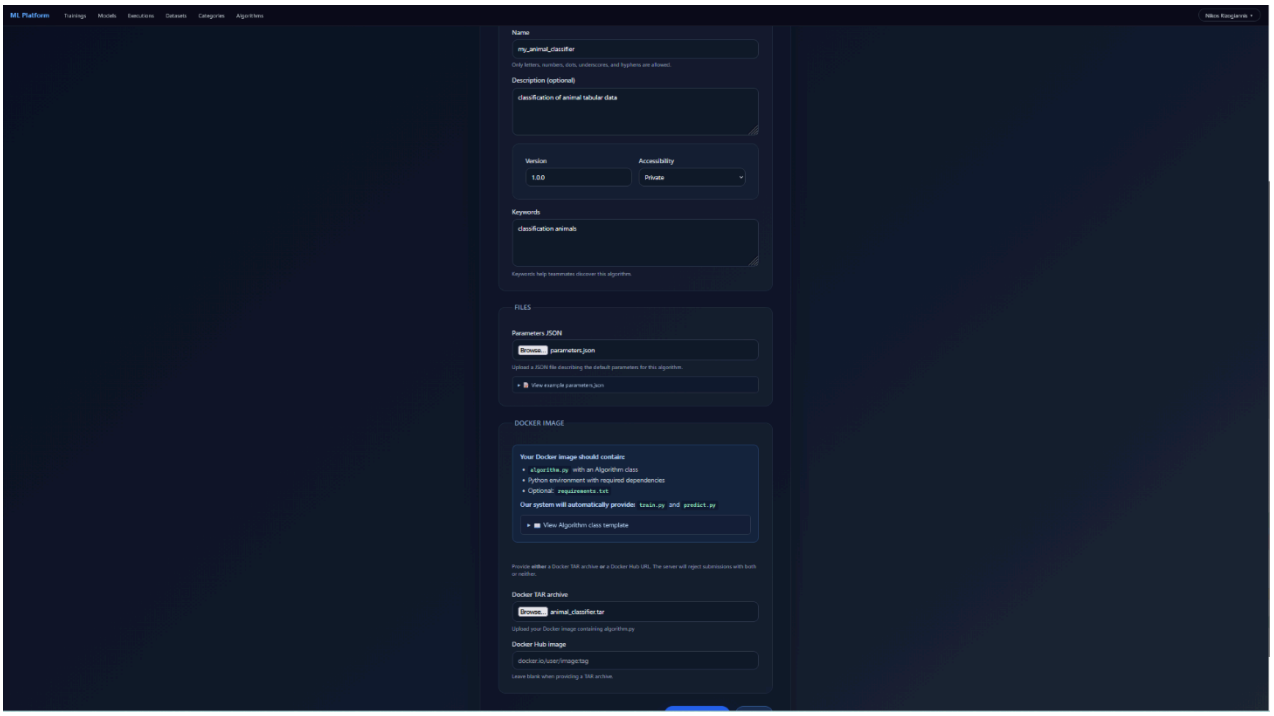
Κατά τη μεταγενέστερη εκπαίδευση μοντέλου με προσαρμοσμένο αλγόριθμο, ο χρήστης δύναται κατ' επιλογή να ορίσει υπερπαραμέτρους μέσω αρχείου JSON, οι οποίες μεταβιβάζονται δυναμικά στο περιβάλλον εκτέλεσης του container, επιτρέποντας την παραμετροποίηση της συμπεριφοράς του αλγορίθμου χωρίς τροποποίηση του κώδικα.



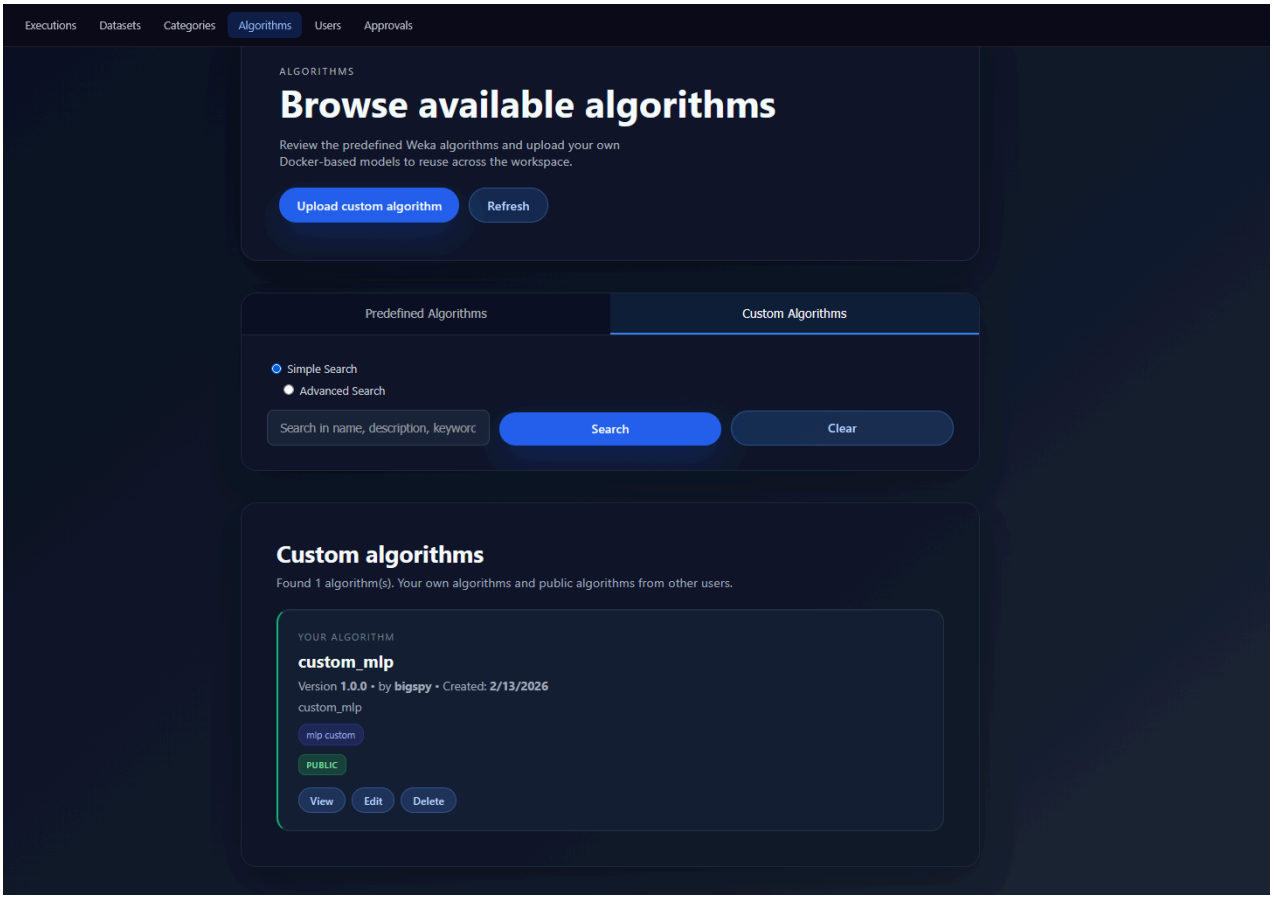
Εικόνα 5.8: Μενού Πλοήγησης - Επιλογή Αλγορίθμων



Εικόνα 5.9: Σελίδα Αλγορίθμων



Εικόνα 5.10: Ανέβασμα Προσαρμοσμένου Αλγορίθμου



Εικόνα 5.11: Πάνελ Προσαρμοσμένων Αλγορίθμων

5.2.3 Σενάριο 3: Εκπαίδευση με Προσαρμοσμένο Αλγόριθμο (Docker-based)

Στο σενάριο αυτό παρουσιάζεται η διαδικασία εκπαίδευσης μοντέλου χρησιμοποιώντας αλγόριθμο που έχει υλοποιηθεί και πακεταριστεί από τον χρήστη σε Docker image. Σε αντίθεση με τους προκαθορισμένους αλγορίθμους Weka, η εκτέλεση βασίζεται εξ ολοκλήρου στον κώδικα που παρέχει ο χρήστης.

Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος, έχει μεταφορτώσει τουλάχιστον ένα σύνολο δεδομένων τύπου Training (βλ. Σενάριο 1) και έχει δημιουργήσει τουλάχιστον έναν προσαρμοσμένο αλγόριθμο (βλ. Σενάριο 2).

Βήματα:

1 Ο χρήστης επιλέγει το στοιχείο Trainings (δείτε Εικόνα 5.12) από το κεντρικό μενού πλοήγησης. Εμφανίζεται η σελίδα διαχείρισης εκπαιδεύσεων.

2 Ο χρήστης πατά το κουμπί Start Training και από τον διάλογο επιλογής τύπου εκπαίδευσης επιλέγει Custom Training, μεταβαίνοντας στη φόρμα εκπαίδευσης με προσαρμοσμένο αλγόριθμο (δείτε Εικόνα 5.13).

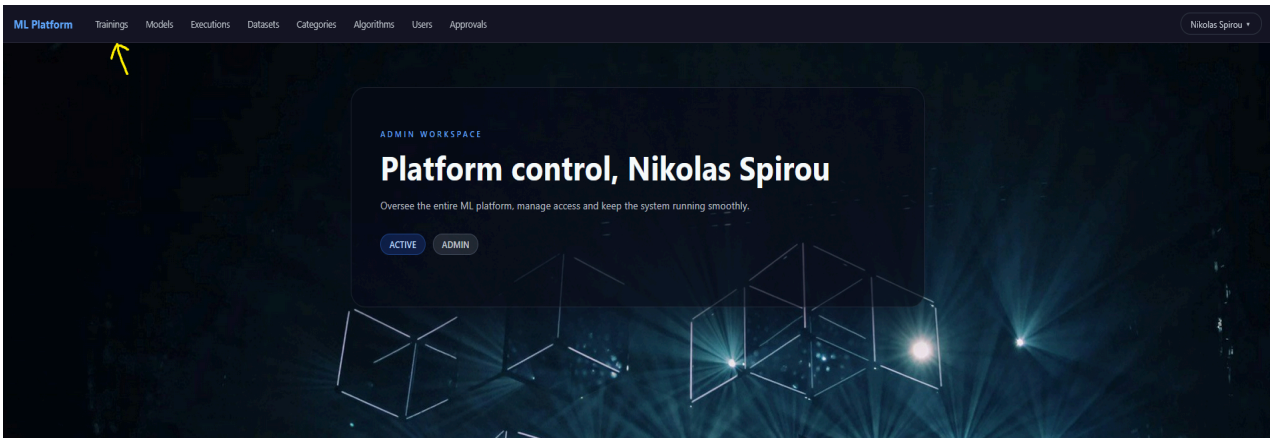
3 Στη φόρμα, ο χρήστης επιλέγει τον προσαρμοσμένο αλγόριθμο από τη λίστα των διαθέσιμων αλγορίθμων (δείτε Εικόνα 5.14), η οποία περιλαμβάνει τόσο τους ιδιωτικούς αλγορίθμους του χρήστη όσο και τους δημόσιους αλγορίθμους άλλων χρηστών.

4 Ο χρήστης επιλέγει ένα υπάρχον σύνολο δεδομένων από τη λίστα των ήδη μεταφορτωμένων συνόλων και, προαιρετικά, μεταφορτώνει ένα αρχείο υπερπαραμέτρων σε μορφή JSON για τη διαμόρφωση της εκπαιδευτικής διαδικασίας (δείτε Εικόνα 5.15).

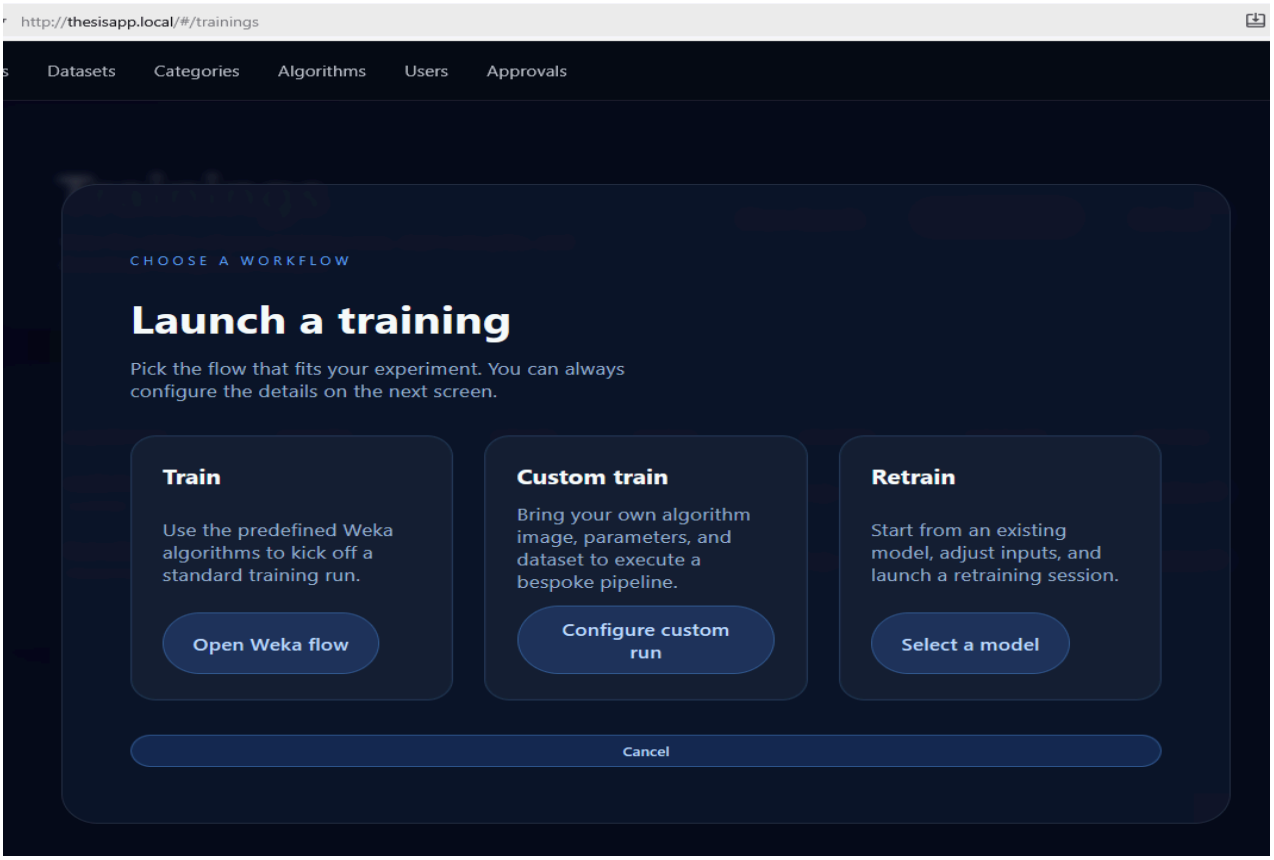
5 Ο χρήστης διαμορφώνει τον ρόλο των στηλών του συνόλου δεδομένων, επιλέγοντας τα χαρακτηριστικά εισόδου (features) και τη μεταβλητή-στόχο (target), και πατά το κουμπί Start Training για την εκκίνηση της εκπαίδευσης (δείτε Εικόνα 5.16).

6 Με την εκκίνηση, η εφαρμογή δημιουργεί ένα Kubernetes Job, το οποίο εκτελεί το Docker image του προσαρμοσμένου αλγορίθμου σε απομονωμένο container. Οι υπερπαράμετροι, το σύνολο δεδομένων και η διαμόρφωση στηλών μεταβιβάζονται δυναμικά στο περιβάλλον εκτέλεσης. Ο χρήστης παρακολουθεί την κατάσταση της εργασίας μέσω του πάνελ ενεργών εργασιών (δείτε Εικόνα 5.17), το οποίο ενημερώνεται αυτόματα κάθε 3 δευτερόλεπτα.

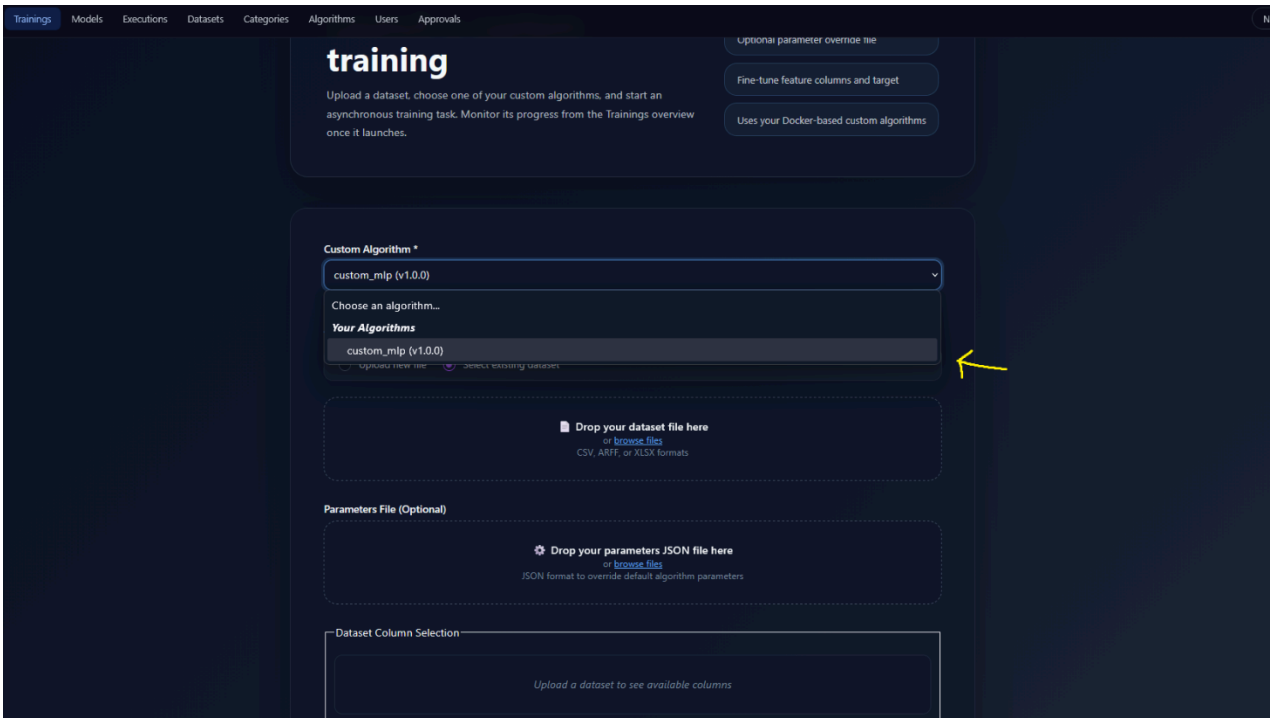
7 Με την ολοκλήρωση, το εκπαιδευμένο μοντέλο αποθηκεύεται στο σύστημα (δείτε Εικόνα 5.19). Ο χρήστης μπορεί να μεταβεί στη σελίδα Trainings (δείτε Εικόνα 5.18) για να επιβεβαιώσει την επιτυχή ολοκλήρωση και να κατεβάσει το παραγόμενο μοντέλο(δείτε Εικόνα 5.20).



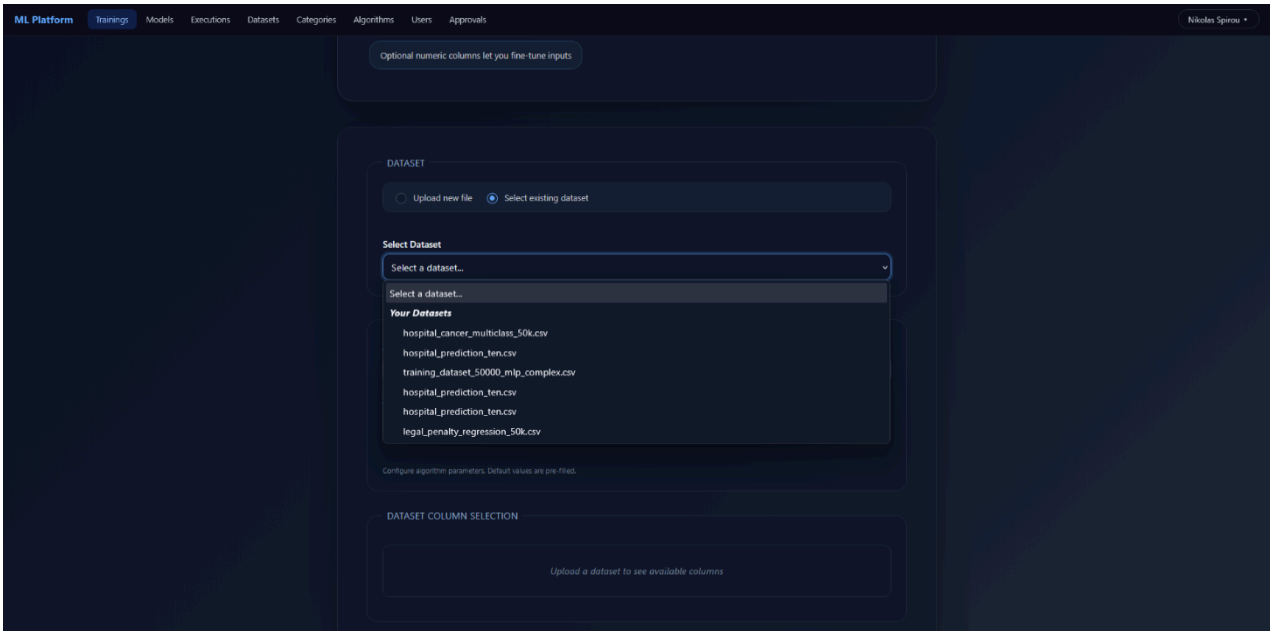
Εικόνα 5.12: Μενού Πλοήγησης - Επιλογή Εκπαιδεύσεων



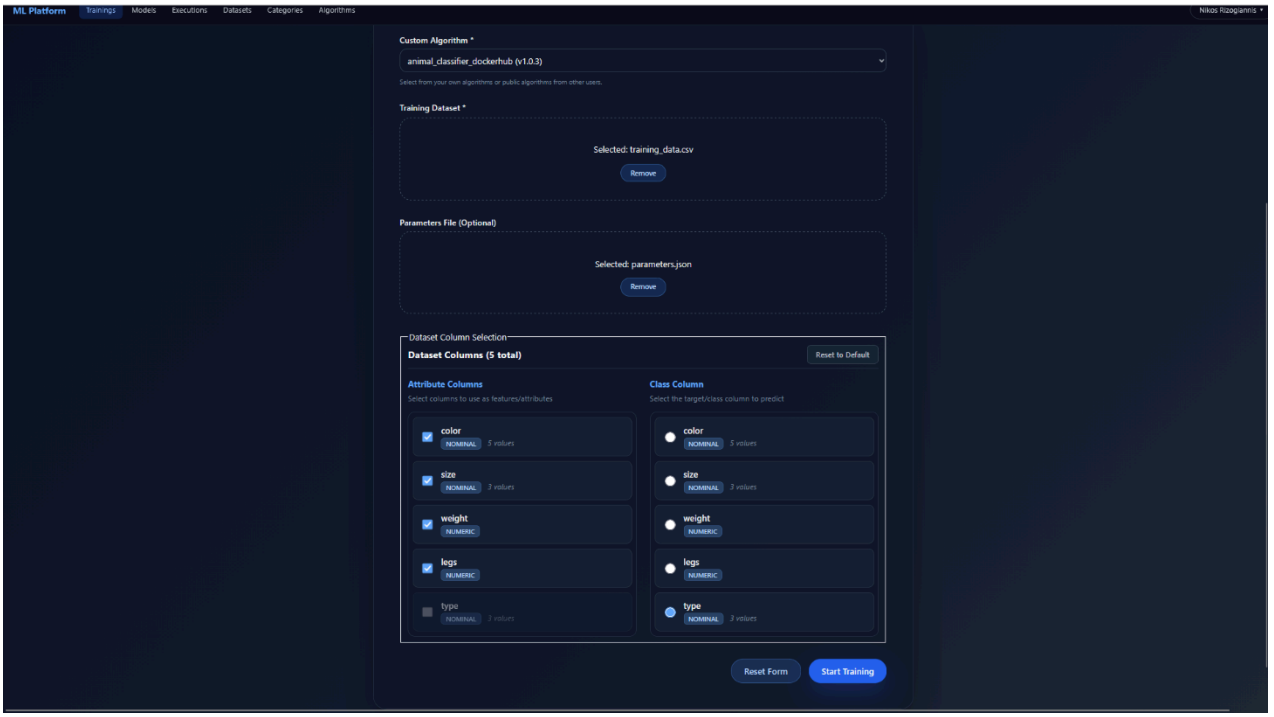
Εικόνα 5.13: Σελίδα Επιλογής Είδους Εκπαίδευσης



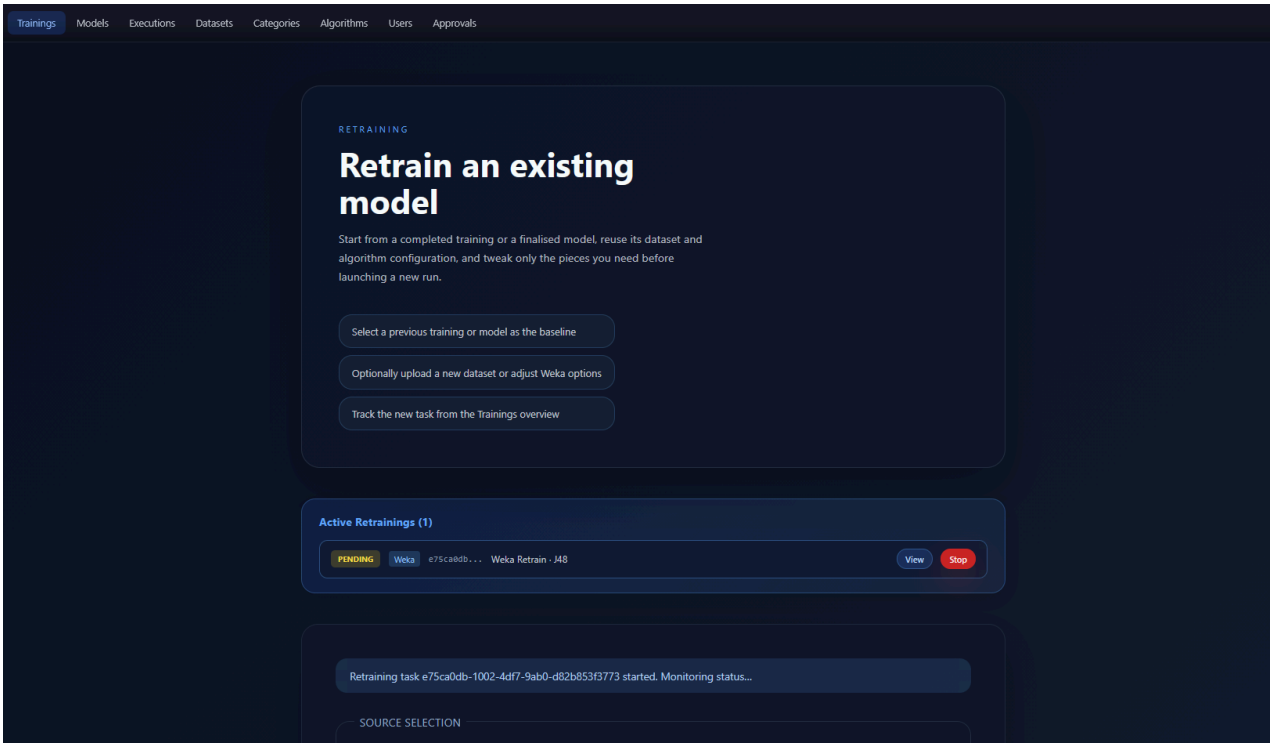
Εικόνα 5.14: Επιλογή Προσαρμοσμένου Αλγορίθμου



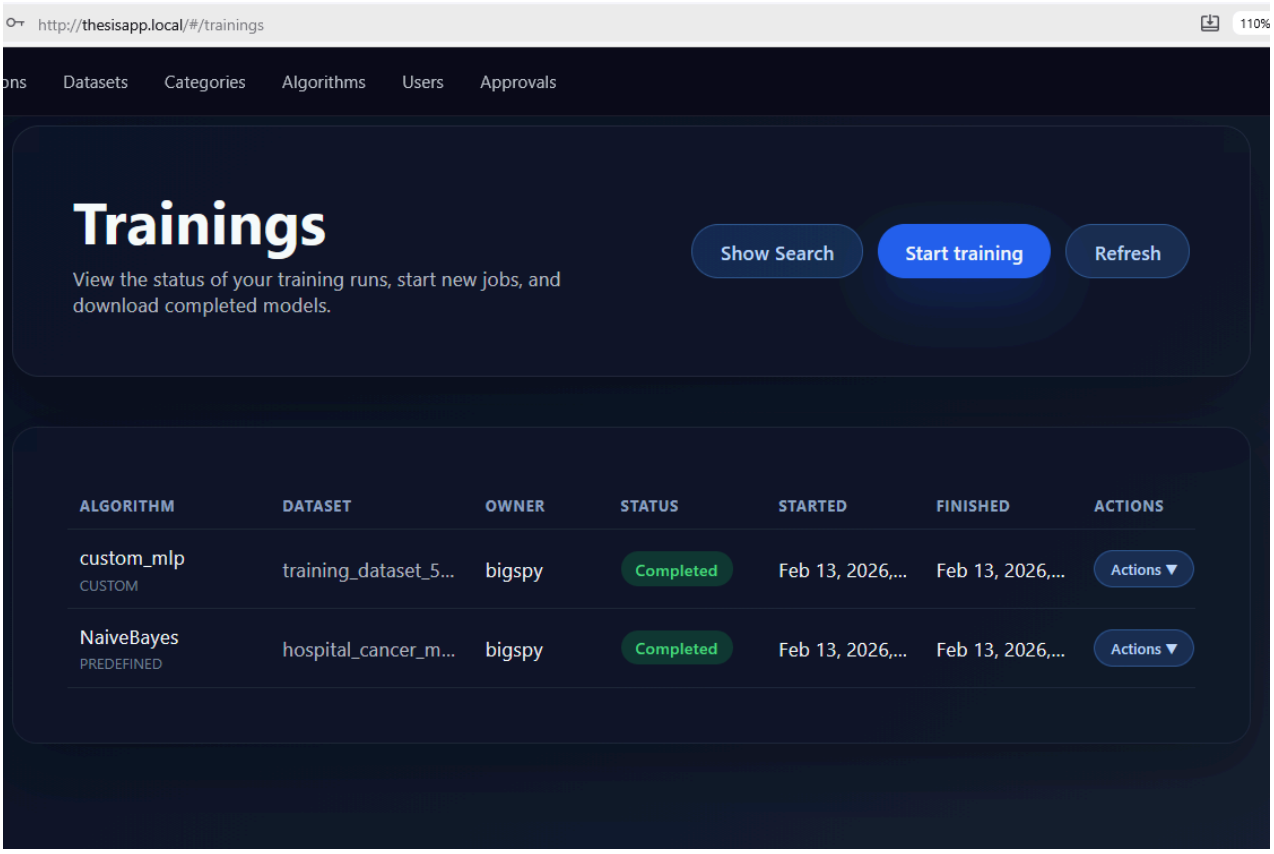
Εικόνα 5.15: Επιλογή Συνόλου Δεδομένων από τα Υπάρχοντα



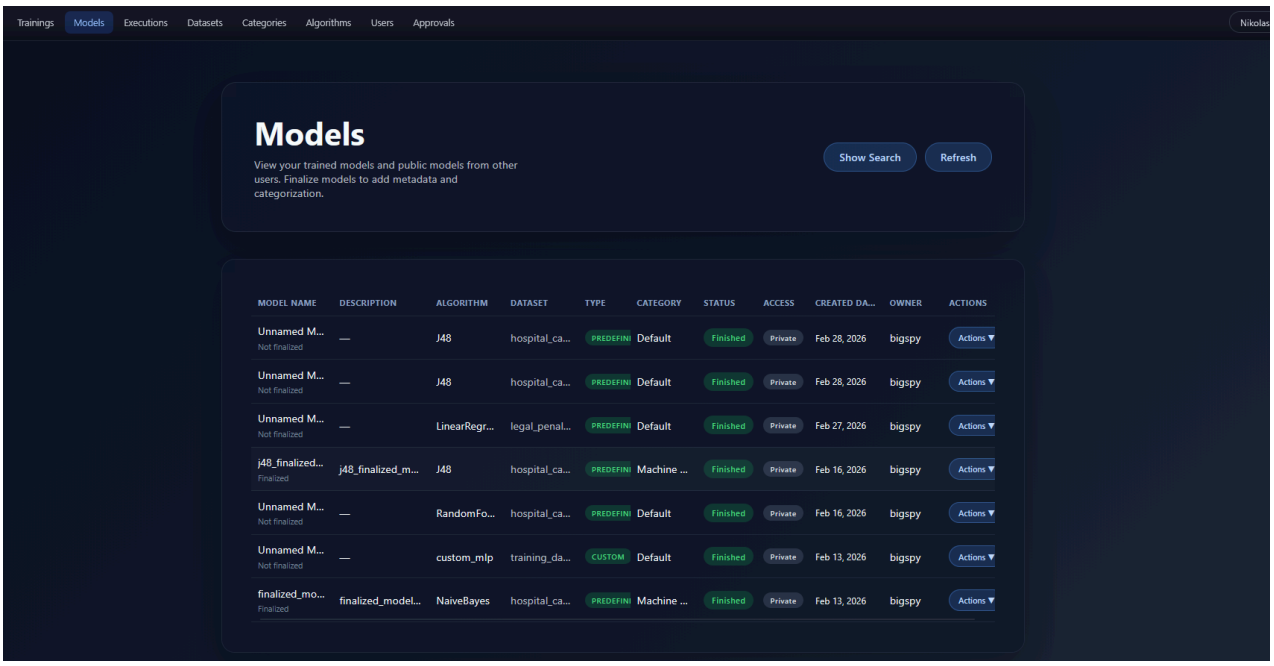
Εικόνα 5.16: Εκκίνηση Εκπαίδευσης Προσαρμοσμένου Αλγορίθμου



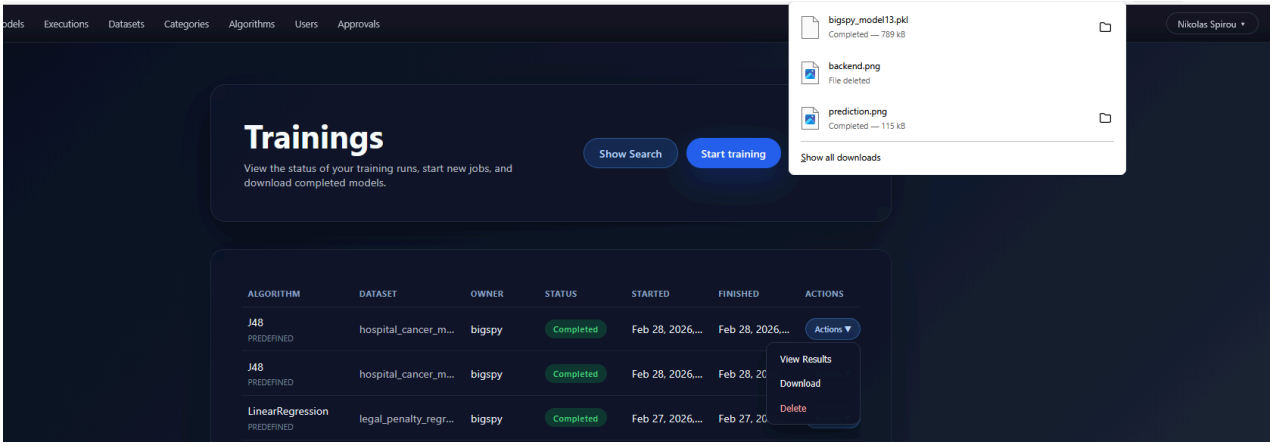
Εικόνα 5.17: Πάνελ Ενεργών Επανεκπαιδεύσεων



Εικόνα 5.18: Λίστα Εκπαιδεύσεων



Εικόνα 5.19: Λίστα Εκπαιδευμένων Μοντέλων



Εικόνα 5.20: Λήψη Εκπαιδευμένου Μοντέλου

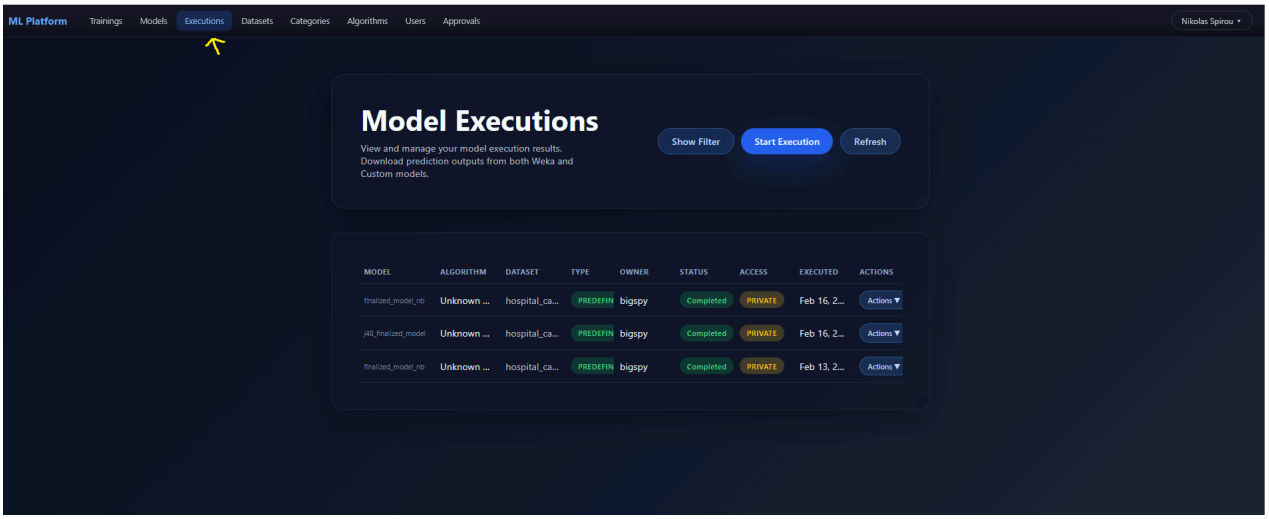
5.2.4 Σενάριο 4: Εκτέλεση Πρόβλεψης με Εκπαιδευμένο Μοντέλο

Στο σενάριο αυτό παρουσιάζεται η διαδικασία εκτέλεσης πρόβλεψης σε νέα δεδομένα χρησιμοποιώντας ένα ήδη εκπαιδευμένο μοντέλο. Η πρόβλεψη εκτελείται ασύγχρονα μέσω Kubernetes Job, ακολουθώντας την ίδια αρχιτεκτονική με τη διαδικασία εκπαίδευσης.

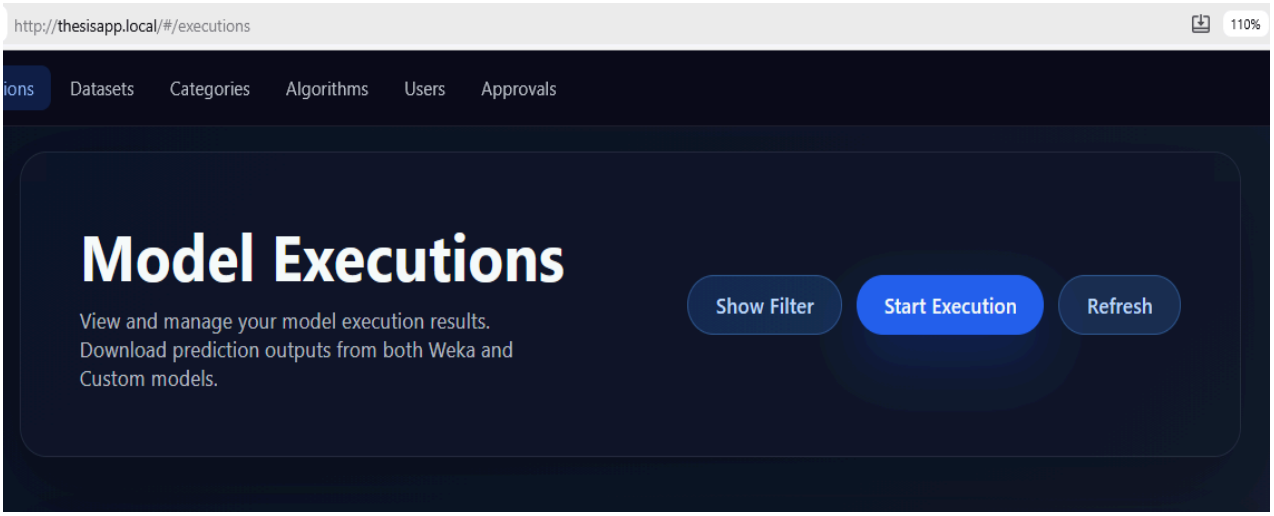
Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος, διαθέτει τουλάχιστον ένα εκπαιδευμένο μοντέλο (βλ. Σενάριο 1 ή 3) και έχει μεταφορτώσει ένα σύνολο δεδομένων τύπου Prediction που περιέχει τα νέα δεδομένα προς πρόβλεψη.

Βήματα:

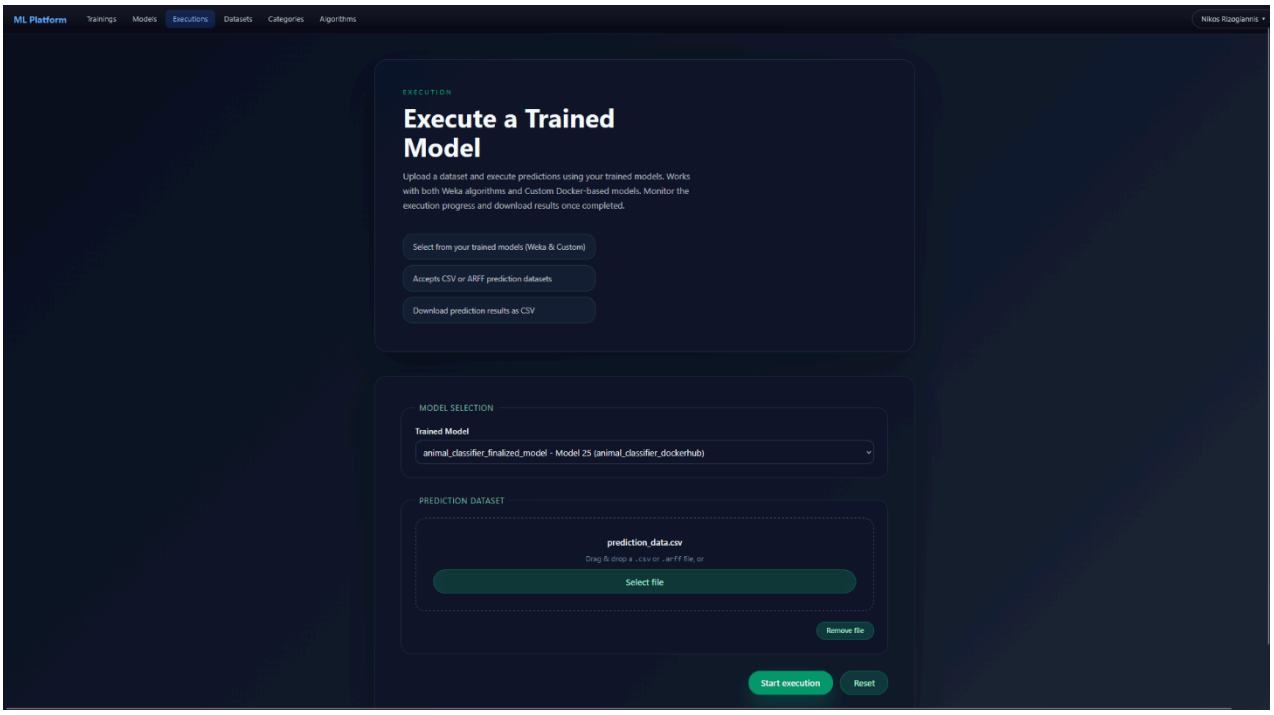
1. Ο χρήστης επιλέγει το στοιχείο Executions από το κεντρικό μενού πλοήγησης και εμφανίζεται η σελίδα ιστορικού εκτελέσεων (δείτε Εικόνα 5.21), η οποία περιλαμβάνει τη λίστα προηγούμενων προβλέψεων και το κουμπί εκκίνησης νέας εκτέλεσης.
2. Ο χρήστης πατά το κουμπί Start Execution (δείτε Εικόνα 5.22) και μεταβαίνει στη φόρμα εκτέλεσης πρόβλεψης (δείτε Εικόνα 5.23).
3. Στη φόρμα, ο χρήστης επιλέγει το εκπαιδευμένο μοντέλο από τη λίστα των διαθέσιμων μοντέλων (δείτε Εικόνα 5.24).
4. Ο χρήστης επιλέγει το σύνολο δεδομένων πρόβλεψης από τη λίστα των ήδη μεταφορτωμένων συνόλων (δείτε Εικόνα 5.25) και πατά το κουμπί Start Execution για την εκκίνηση της πρόβλεψης.
5. Η εφαρμογή δημιουργεί ένα Kubernetes Job, το οποίο φορτώνει το εκπαιδευμένο μοντέλο και εφαρμόζει τον αλγόριθμο πρόβλεψης στα νέα δεδομένα. Ο χρήστης παρακολουθεί την κατάσταση της εργασίας μέσω του πάνελ ενεργών εργασιών (δείτε Εικόνα 5.26), το οποίο ενημερώνεται αυτόματα κάθε 3 δευτερόλεπτα.
6. Με την ολοκλήρωση, τα αποτελέσματα της πρόβλεψης αποθηκεύονται σε μορφή CSV. Ο χρήστης μεταβαίνει στη σελίδα Executions και επιλέγει Download (δείτε Εικόνα 5.27) για την ολοκληρωμένη εκτέλεση, οπότε λαμβάνει το αρχείο αποτελεσμάτων με τις τιμές πρόβλεψης για κάθε εγγραφή του συνόλου δεδομένων.



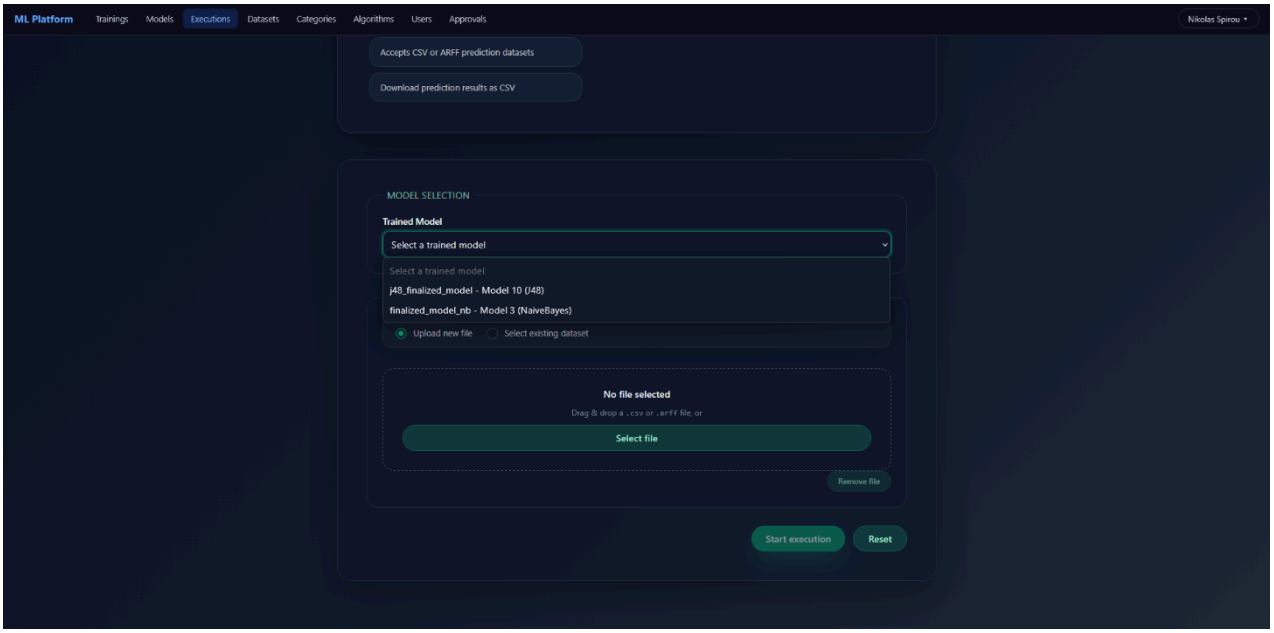
Εικόνα 5.21: Μενού Πλοήγησης - Επιλογή Εκτελέσεων



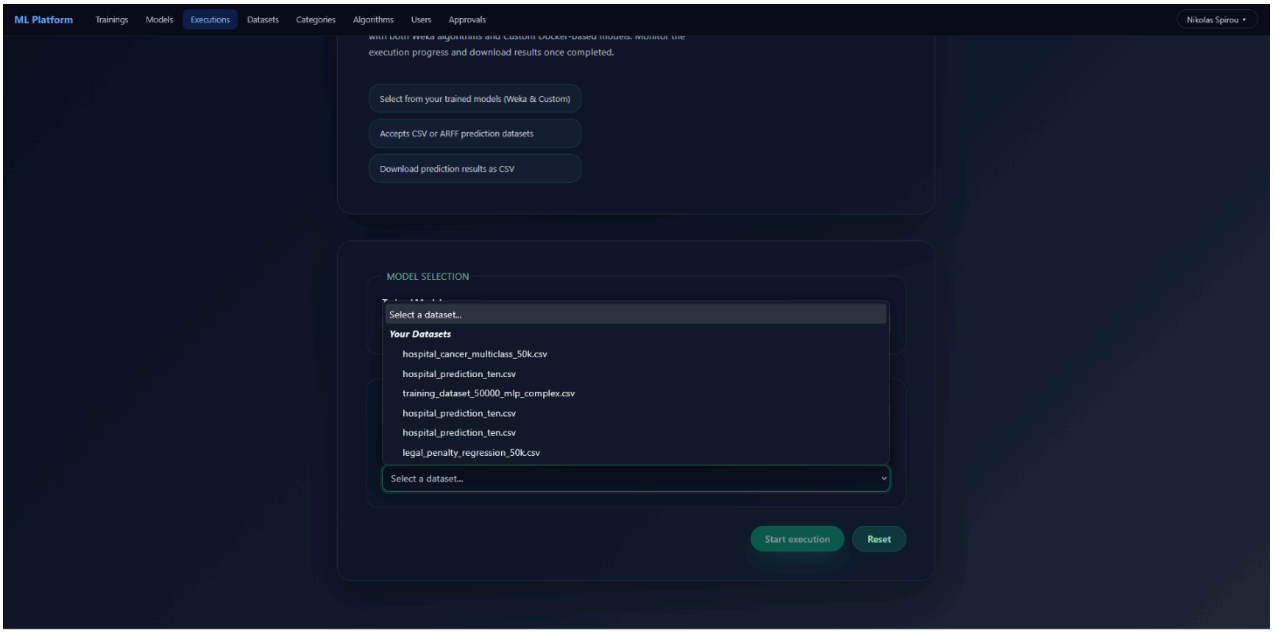
Εικόνα 5.22: Σελίδα Επιλογών Εκτέλεσης



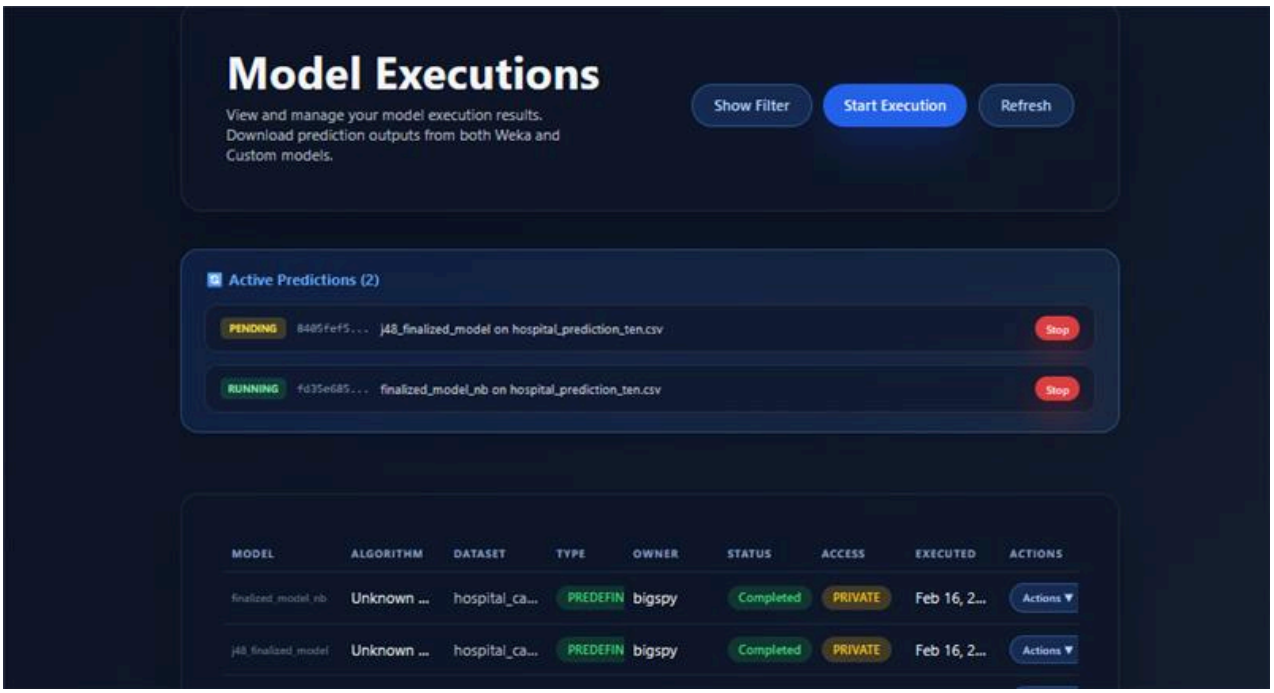
Εικόνα 5.23: Εκτέλεση Προσαρμοσμένου Μοντέλου



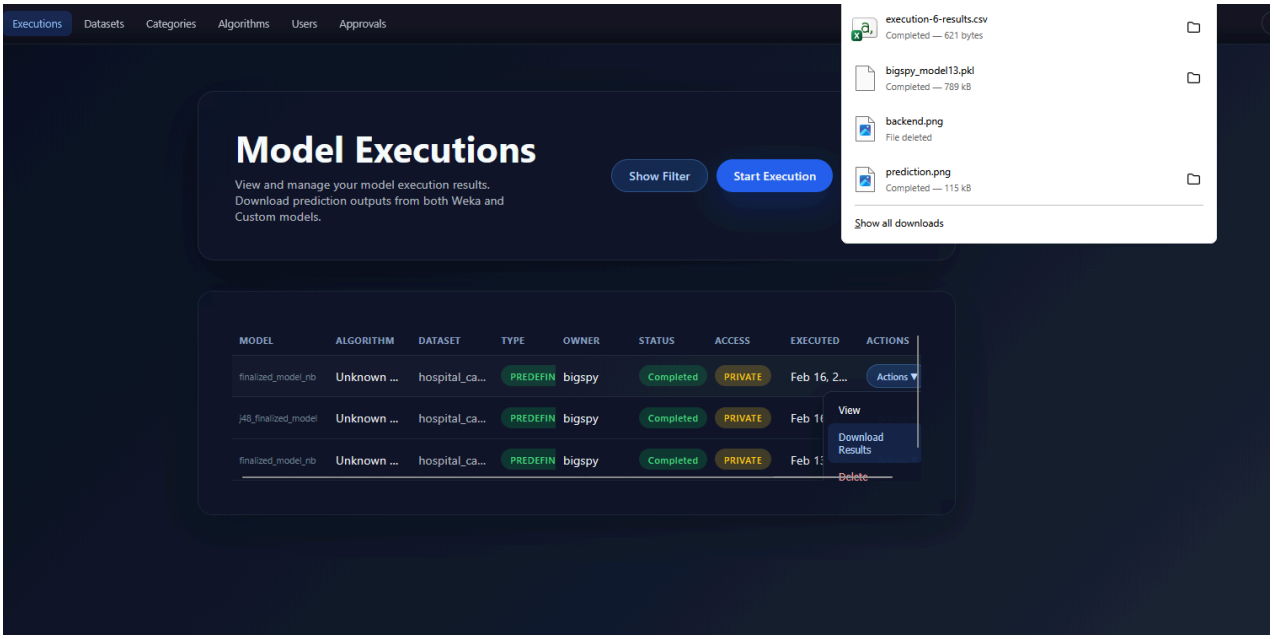
Εικόνα 5.24: Επιλογή Εκπαιδευμένου Μοντέλου



Εικόνα 5.25: Επιλογή Συνόλου Δεδομένων Πρόβλεψης από τα Υπάρχοντα



Εικόνα 5.26: Ενεργές Εκτελέσεις Μοντέλων



Εικόνα 5.27: Λήψη Αποτελεσμάτων Εκτέλεσης

5.2.5 Σενάριο 5: Διαχείριση Λογαριασμού Χρήστη

Στο σενάριο αυτό παρουσιάζονται οι βασικές λειτουργίες διαχείρισης λογαριασμού χρήστη, οι οποίες αποτελούν προαπαιτούμενο για τη χρήση όλων των λειτουργιών της εφαρμογής.

1) Εγγραφή νέου χρήστη

Προϋποθέσεις: Ο χρήστης δεν διαθέτει λογαριασμό στο σύστημα.

Βήματα:

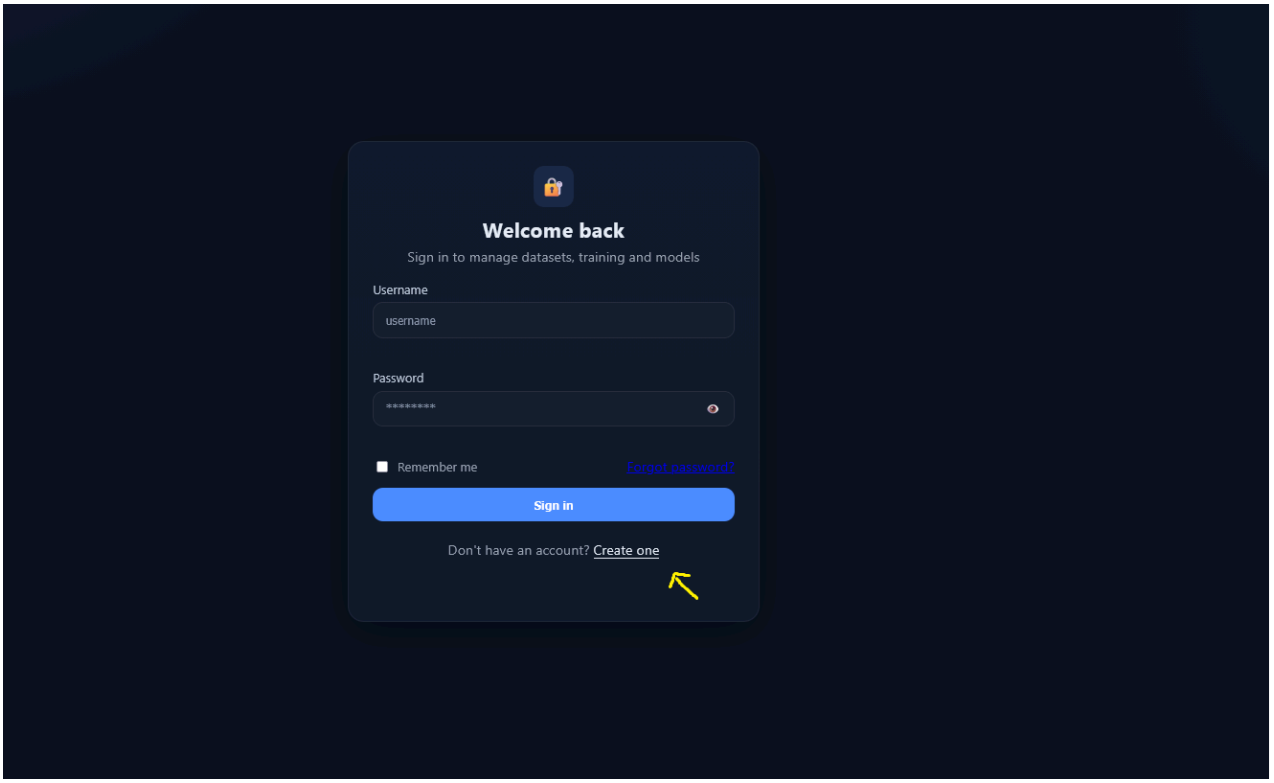
1. Ο χρήστης μεταβαίνει στη σελίδα εγγραφής επιλέγοντας Register από την αρχική σελίδα της εφαρμογής (δείτε Εικόνα 5.28).
2. Στη φόρμα εγγραφής (δείτε Εικόνα 5.29), ο χρήστης συμπληρώνει τα απαιτούμενα στοιχεία: όνομα χρήστη, email, κωδικό πρόσβασης, όνομα, επώνυμο, ηλικία, επάγγελμα και χώρα.
3. Με την υποβολή, ο λογαριασμός δημιουργείται και ο χρήστης μπορεί πλέον να συνδεθεί στο σύστημα.

2) Αυθεντικοποίηση (Σύνδεση)

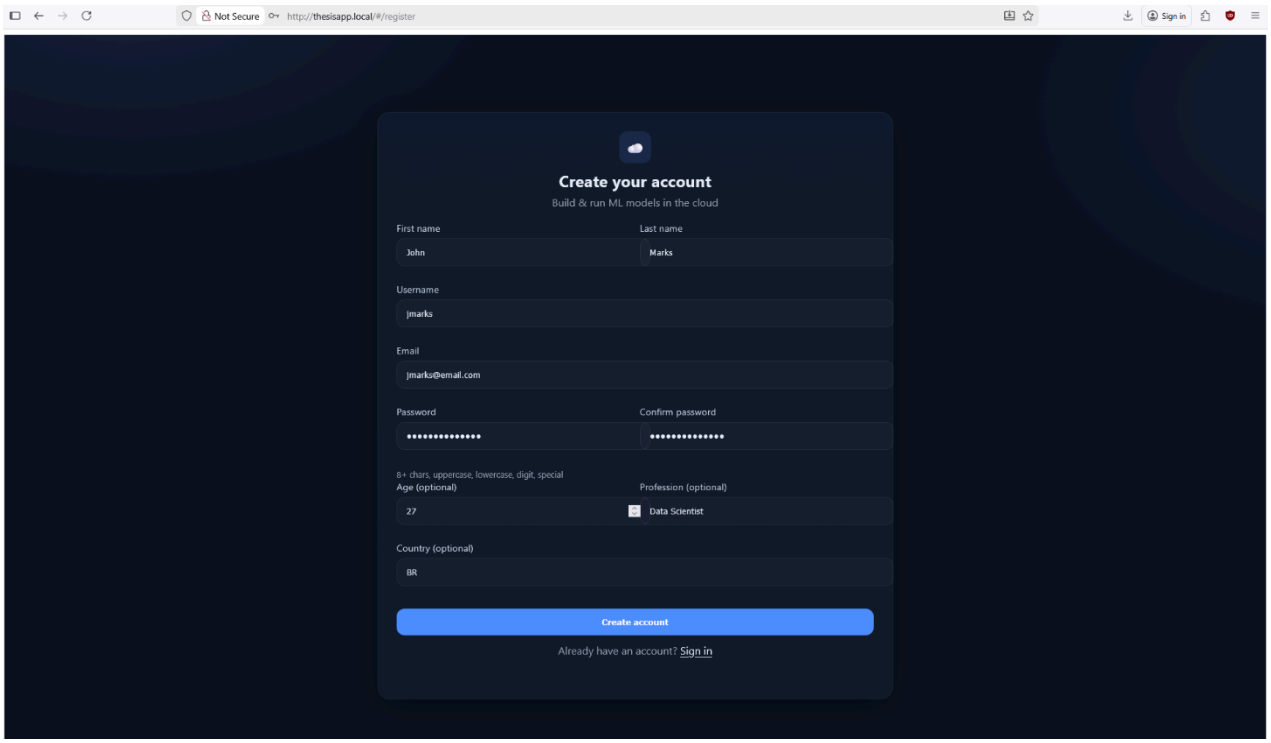
Προϋποθέσεις: Ο χρήστης διαθέτει ενεργό λογαριασμό στο σύστημα.

Βήματα:

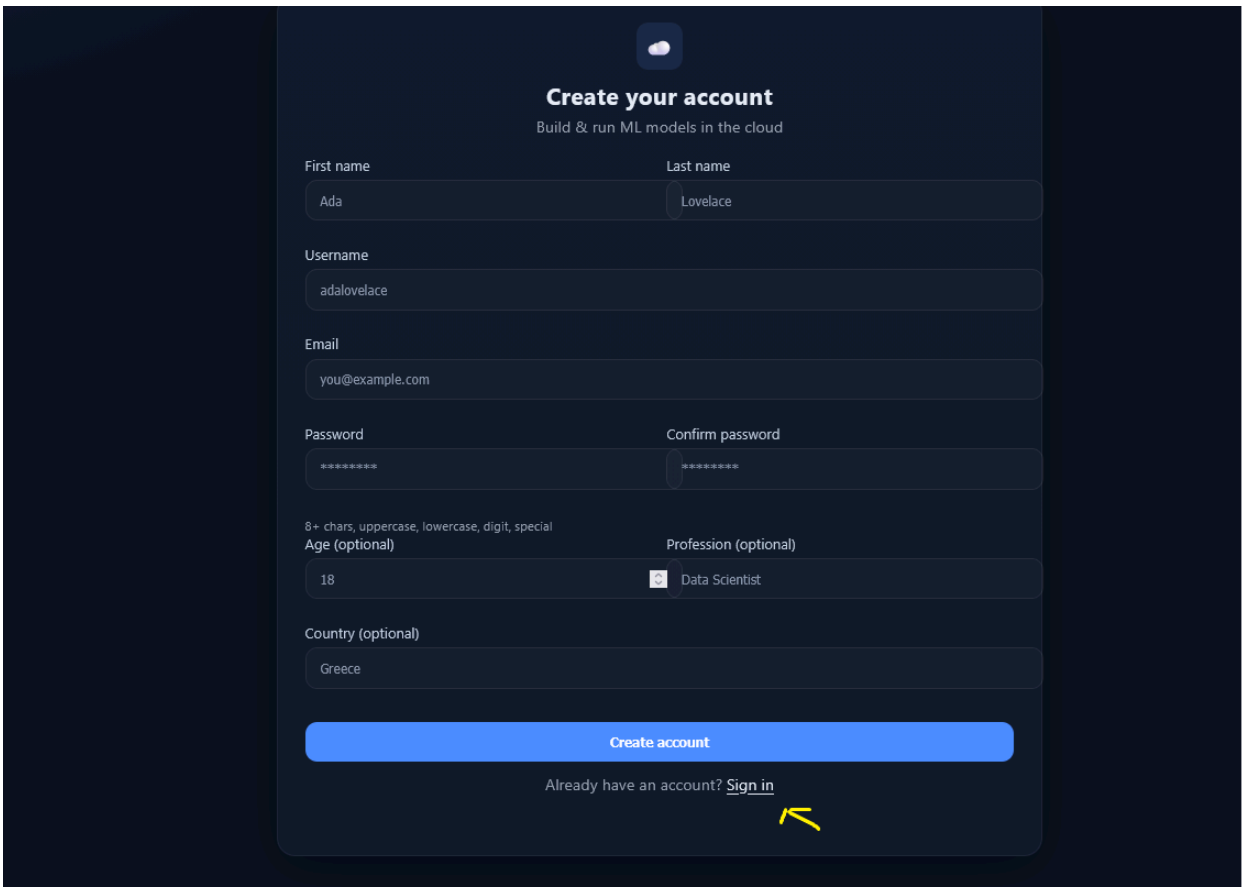
1. Ο χρήστης μεταβαίνει στη σελίδα σύνδεσης επιλέγοντας Sign In (δείτε Εικόνα 5.30) από την αρχική σελίδα.
2. Εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης και πατά Sign In (δείτε Εικόνα 5.31).
3. Μετά την επιτυχή αυθεντικοποίηση, εκδίδεται ένα JWT token το οποίο αποθηκεύεται τοπικά και χρησιμοποιείται αυτόματα σε κάθε επόμενο αίτημα. Ο χρήστης ανακατευθύνεται στην κεντρική σελίδα της εφαρμογής, όπου εμφανίζεται μήνυμα καλωσορίσματος και ο ρόλος του (δείτε Εικόνα 5.32).



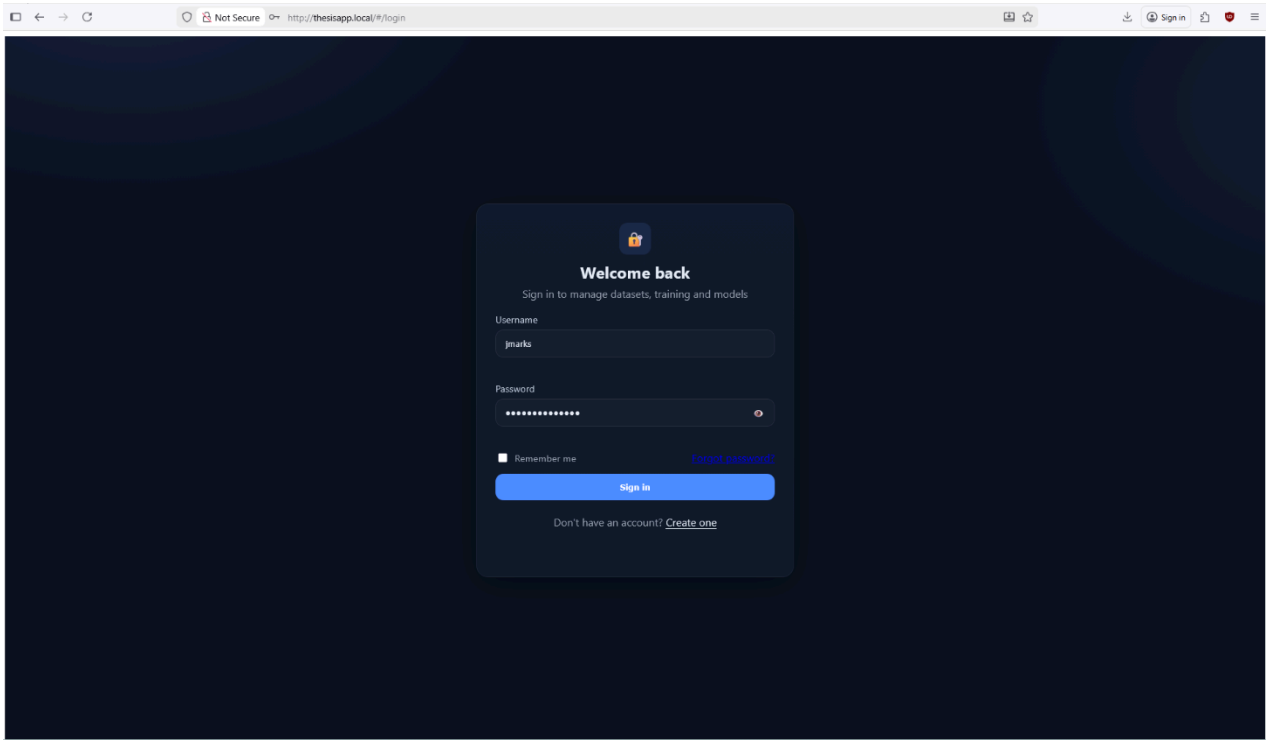
Εικόνα 5.28: Πλοήγηση στην Εγγραφή Χρήστη



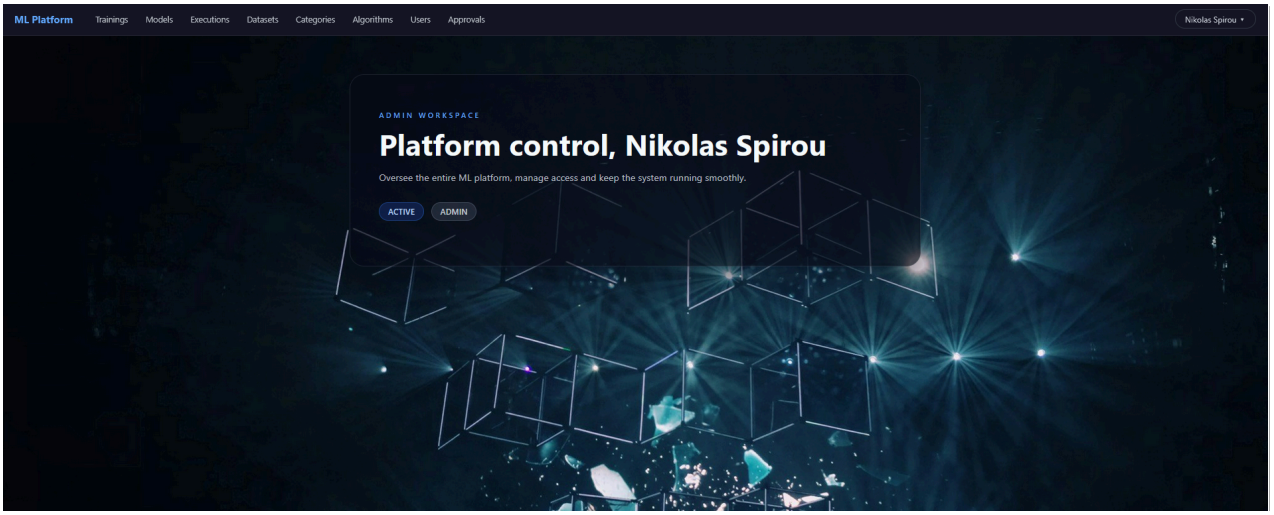
Εικόνα 5.29: Εγγραφή Χρήστη



Εικόνα 5.30: Πλοήγηση στη Σύνδεση Χρήστη



Εικόνα 5.31: Αυθεντικοποίηση Χρήστη



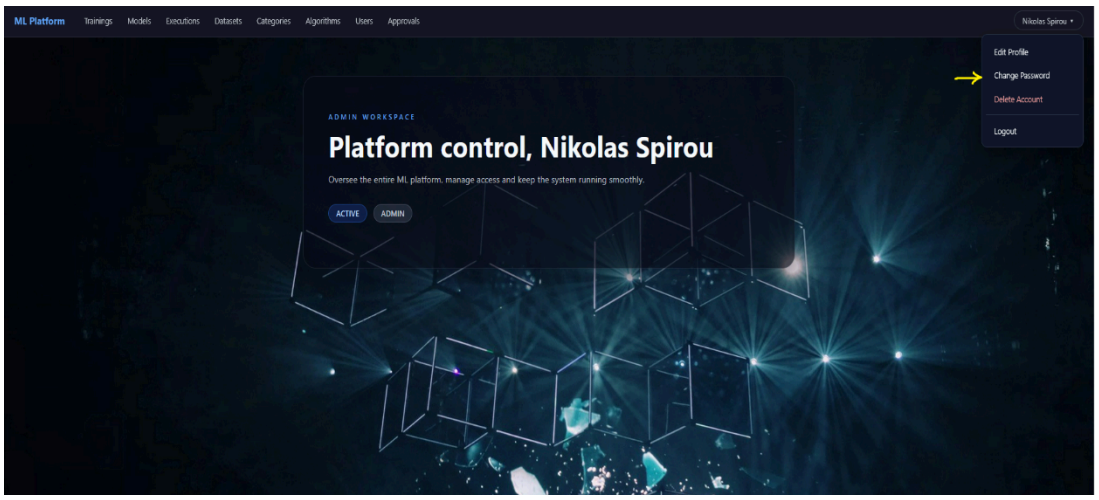
Εικόνα 5.32: Σελίδα Καλωσορίσματος Χρήστη

3) Αλλαγή κωδικού πρόσβασης

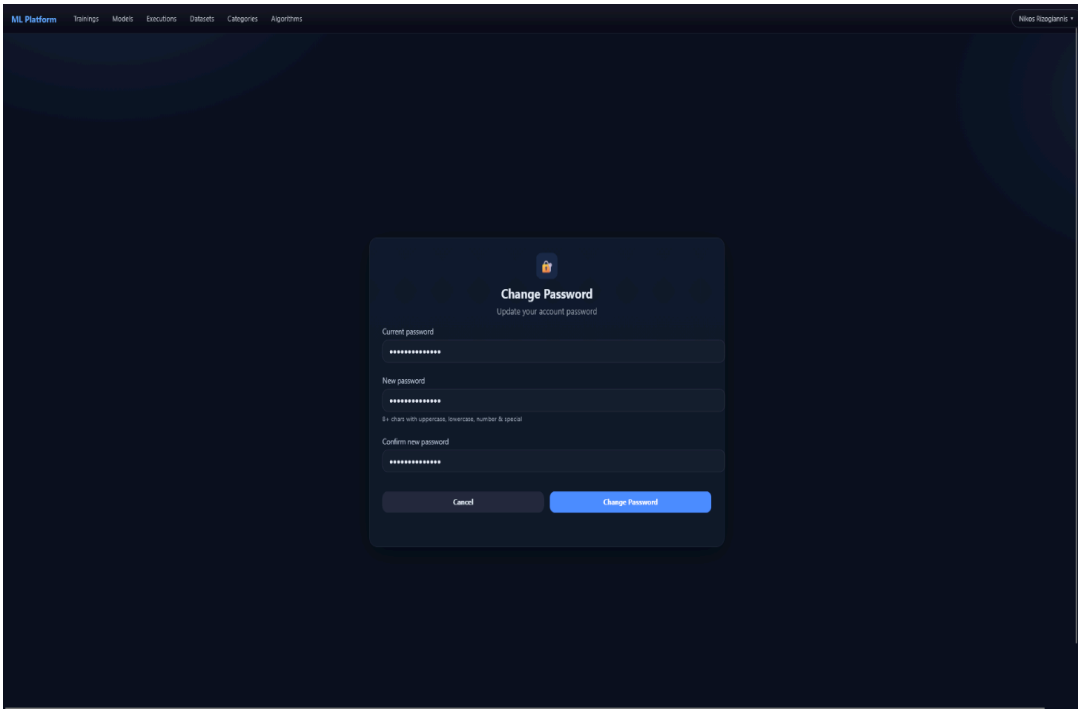
Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος στο σύστημα.

Βήματα:

1. Ο χρήστης μεταβαίνει στη σελίδα του προφίλ του επιλέγοντας Profile από το μενού πλοήγησης και στη συνέχεια επιλέγει Change Password (δείτε Εικόνα 5.33).
2. Στη φόρμα αλλαγής κωδικού, εισάγει τον τρέχοντα κωδικό πρόσβασης και τον νέο κωδικό (δύο φορές για επιβεβαίωση) (δείτε Εικόνα 5.34).
3. Με την υποβολή, ο κωδικός ενημερώνεται και αποθηκεύεται κρυπτογραφημένος με τον αλγόριθμο Argon2id.



Εικόνα 5.33: Πλοήγηση στην Αλλαγή Κωδικού Πρόσβασης



Εικόνα 5.34: Αλλαγή Κωδικού Πρόσβασης

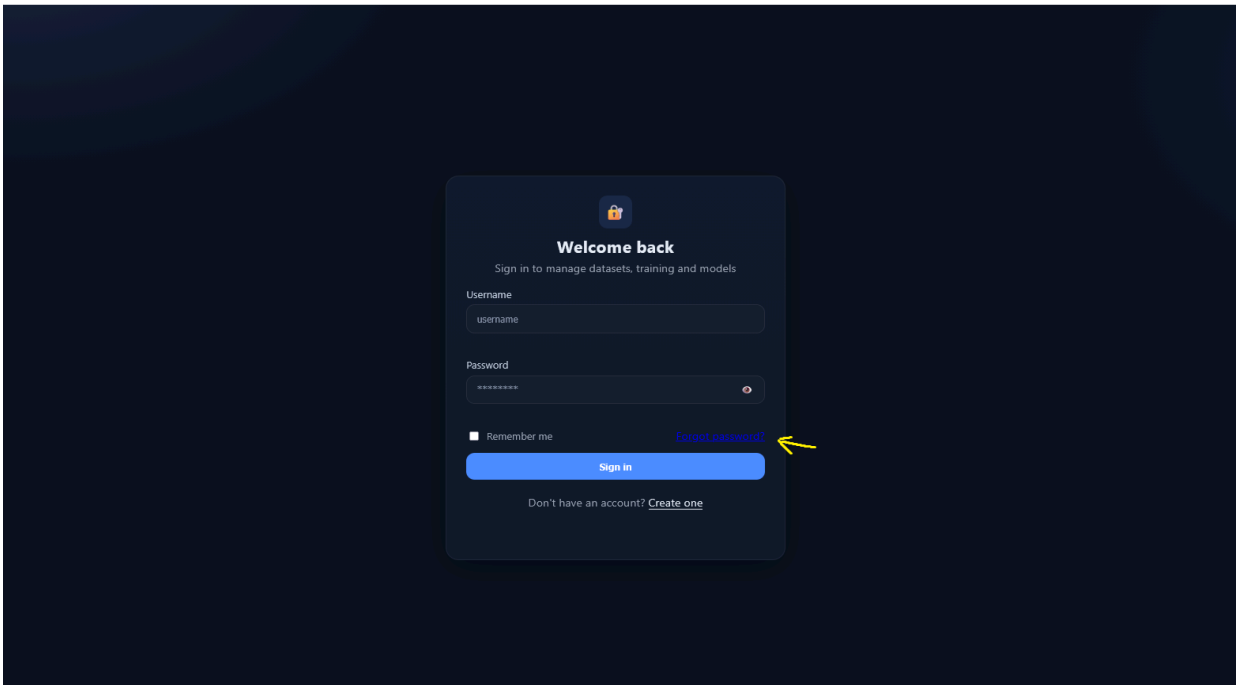
4) Επαναφορά κωδικού πρόσβασης (Reset Password)

Προϋποθέσεις: Ο χρήστης διαθέτει λογαριασμό αλλά δεν θυμάται τον κωδικό πρόσβασής του.

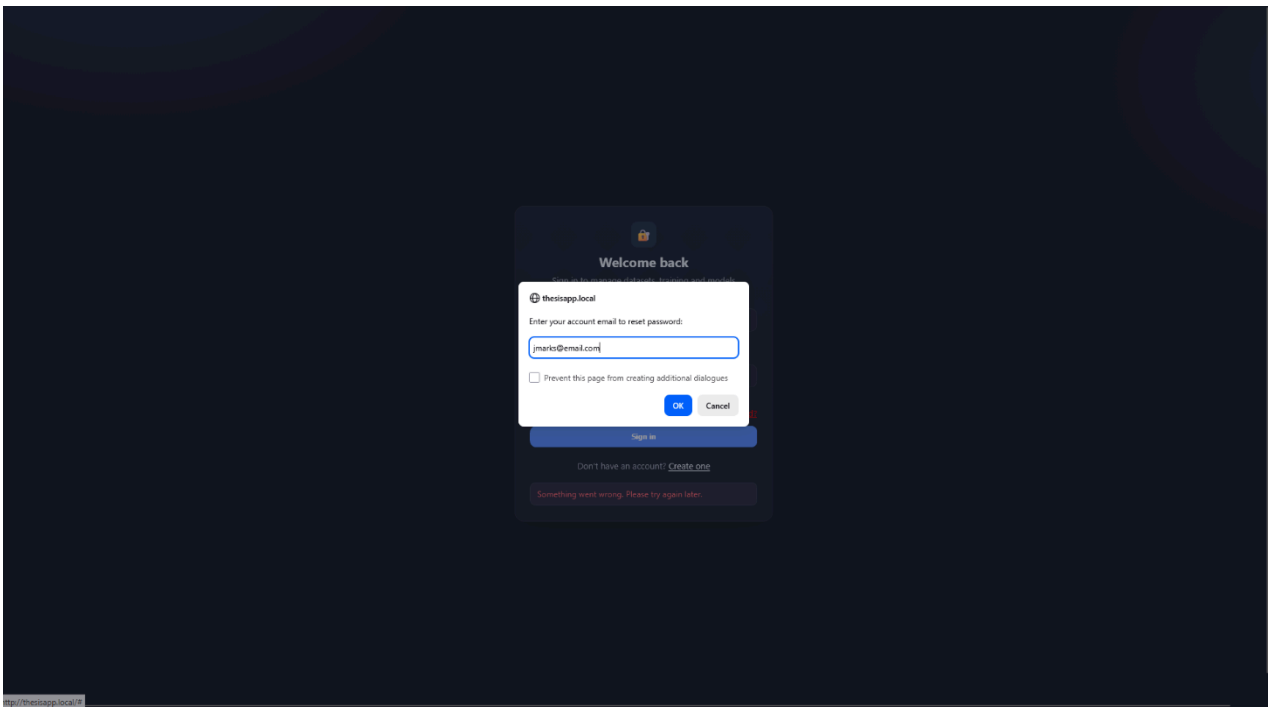
Βήματα:

1. Ο χρήστης επιλέγει Forgot Password από τη σελίδα σύνδεσης (δείτε Εικόνα 5.35).
2. Εισάγει τη διεύθυνση email που συνδέεται με τον λογαριασμό του και υποβάλλει το αίτημα (δείτεΕικόνα 5.36).

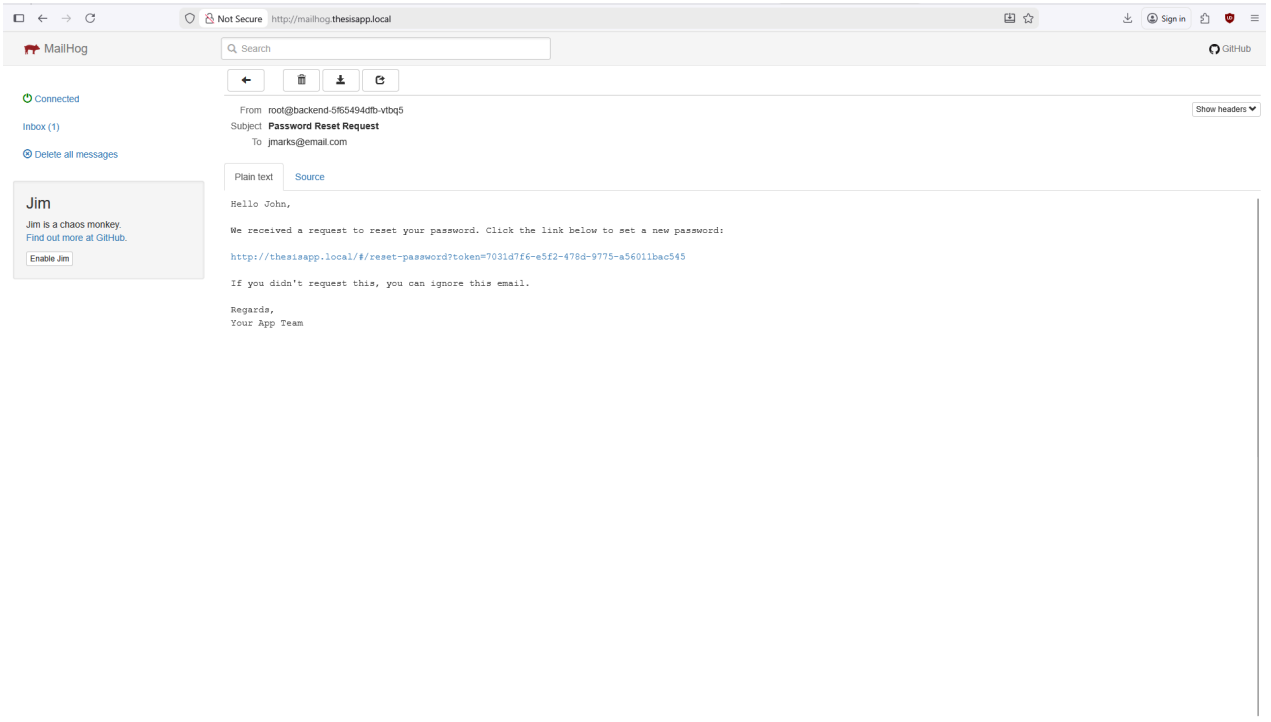
3. Το σύστημα αποστέλλει email με σύνδεσμο επαναφοράς κωδικού μέσω της υπηρεσίας MailHog (δείτε Εικόνα 5.37). Ο χρήστης ακολουθεί τον σύνδεσμο, ορίζει νέο κωδικό πρόσβασης (δείτε Εικόνα 5.38) και μπορεί πλέον να συνδεθεί με τα νέα διαπιστευτήρια .



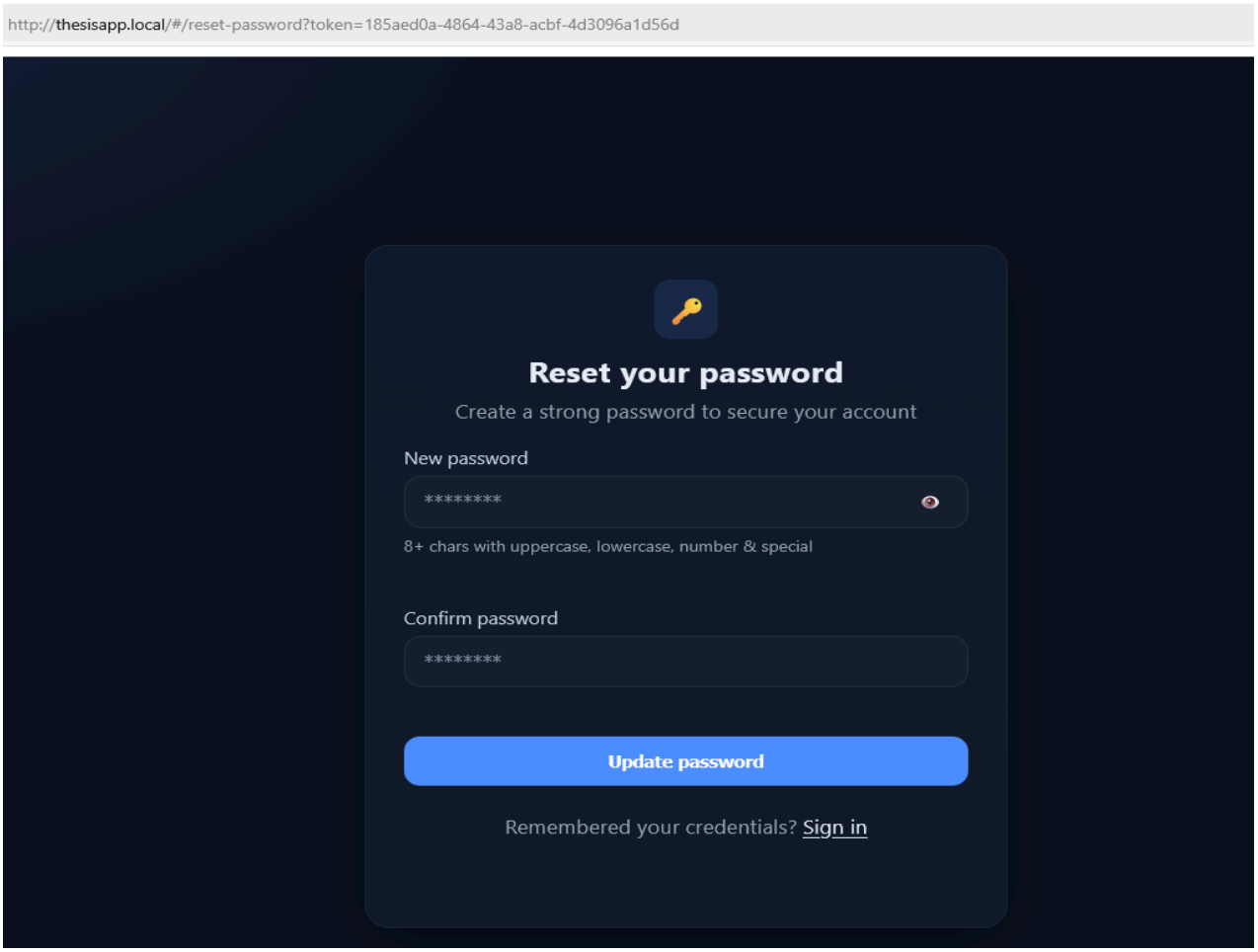
Εικόνα 5.35: Πλοήγηση στην Ανάκτηση Κωδικού Πρόσβασης



Εικόνα 5.36: Αίτηση Επαναφοράς Κωδικού Πρόσβασης



Εικόνα 5.37: Περιεχόμενο Μηνύματος Επαναφοράς Κωδικού



Εικόνα 5.38: Επαναφορά Κωδικού Πρόσβασης

5.2.6 Σενάριο 6: Διαχείριση Κατηγοριών

Στο σενάριο αυτό παρουσιάζεται η διαδικασία δημιουργίας και έγκρισης κατηγοριών, οι οποίες χρησιμοποιούνται για την οργάνωση και ταξινόμηση των μοντέλων και των συνόλων δεδομένων στο σύστημα. Η δημιουργία κατηγοριών ακολουθεί ροή έγκρισης (approval workflow), κατά την οποία ο χρήστης υποβάλλει αίτημα και ο διαχειριστής το εγκρίνει ή το απορρίπτει.

1) Υποβολή αιτήματος κατηγορίας (Χρήστης)

Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος στο σύστημα.

Βήματα:

1. Ο χρήστης επιλέγει το στοιχείο Categories από το κεντρικό μενού πλοήγησης (δείτε Εικόνα 5.39). Εμφανίζεται η σελίδα διαχείρισης κατηγοριών (δείτε Εικόνα 5.40), η οποία περιλαμβάνει τη λίστα των εγκεκριμένων κατηγοριών σε μορφή σελιδοποιημένου πίνακα.

2. Ο χρήστης πατά το κουμπί Propose Category, οπότε εμφανίζεται ο διάλογος υποβολής αιτήματος (δείτε Εικόνα 5.41).
3. Στη φόρμα, ο χρήστης συμπληρώνει το όνομα της κατηγορίας (υποχρεωτικό), την περιγραφή (προαιρετικά) και, εφόσον επιθυμεί, επιλέγει μία ή περισσότερες γονικές κατηγορίες, δημιουργώντας ιεραρχική δομή (δείτε Εικόνα 5.42).
4. Με την υποβολή, δημιουργείται αίτημα κατηγορίας σε κατάσταση Pending και ο χρήστης ενημερώνεται ότι αναμένεται η έγκριση από τον διαχειριστή (δείτε Εικόνα 5.43).

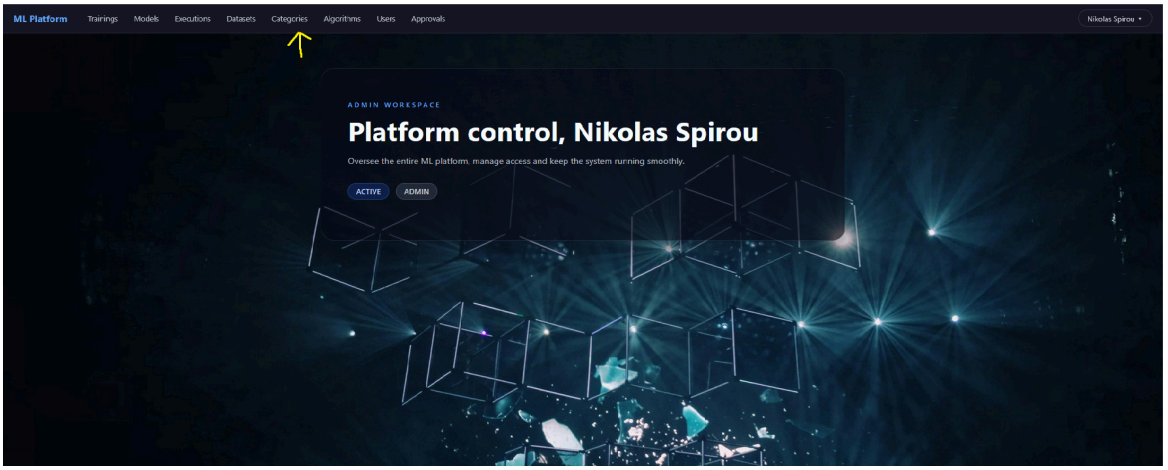
2) Έγκριση ή απόρριψη αιτήματος (Διαχειριστής)

Προϋποθέσεις: Ο χρήστης (διαχειριστής) είναι αυθεντικοποιημένος με ρόλο ADMIN και υπάρχει τουλάχιστον ένα εκκρεμές αίτημα κατηγορίας.

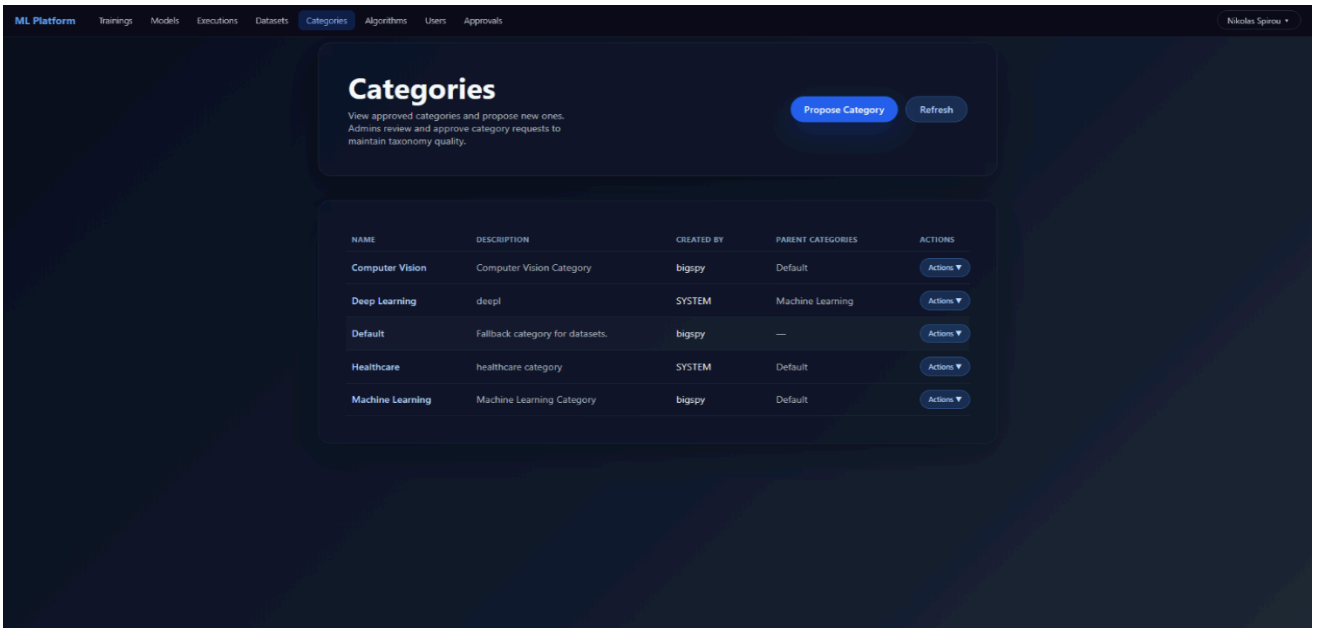
Βήματα:

1. Ο διαχειριστής επιλέγει το στοιχείο Approvals από το κεντρικό μενού πλοήγησης (δείτε Εικόνα 5.44). Εμφανίζεται η σελίδα διαχείρισης αιτημάτων (δείτε Εικόνα 5.43), η οποία περιλαμβάνει στατιστικά (εκκρεμή, εγκεκριμένα, απορριφθέντα) και τη λίστα αιτημάτων με δυνατότητα φιλτραρίσματος κατά κατάσταση.
2. Ο διαχειριστής επιλέγει ένα εκκρεμές αίτημα, οπότε εμφανίζεται παράθυρο (δείτε Εικόνα 5.45) με τα στοιχεία του: όνομα, περιγραφή, αιτών χρήστης, ημερομηνία υποβολής και γονικές κατηγορίες.
3. Ο διαχειριστής επιλέγει μία από τις δύο ενέργειες:
   - Approve: Η κατηγορία δημιουργείται αυτόματα στο σύστημα και είναι πλέον διαθέσιμη σε όλους τους χρήστες (δείτε Εικόνα 5.46).
   - Reject: Ο διαχειριστής εισάγει αιτιολογία απόρριψης και το αίτημα μεταβαίνει σε κατάσταση Rejected (δείτε Εικόνα 5.47).

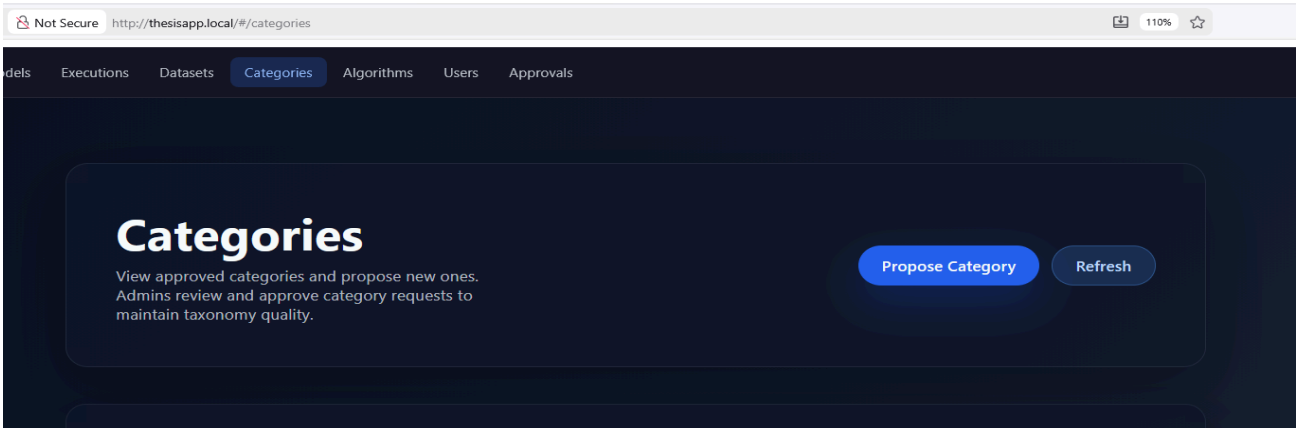
Οι εγκεκριμένες κατηγορίες χρησιμοποιούνται στη συνέχεια κατά την οριστικοποίηση μοντέλων, όπου ο χρήστης υποχρεούται να αντιστοιχίσει το μοντέλο του σε μία κατηγορία. Οι κατηγορίες υποστηρίζουν ιεραρχική δομή μέσω σχέσεων γονέα-παιδιού, επιτρέποντας την οργάνωση σε πολλαπλά επίπεδα.



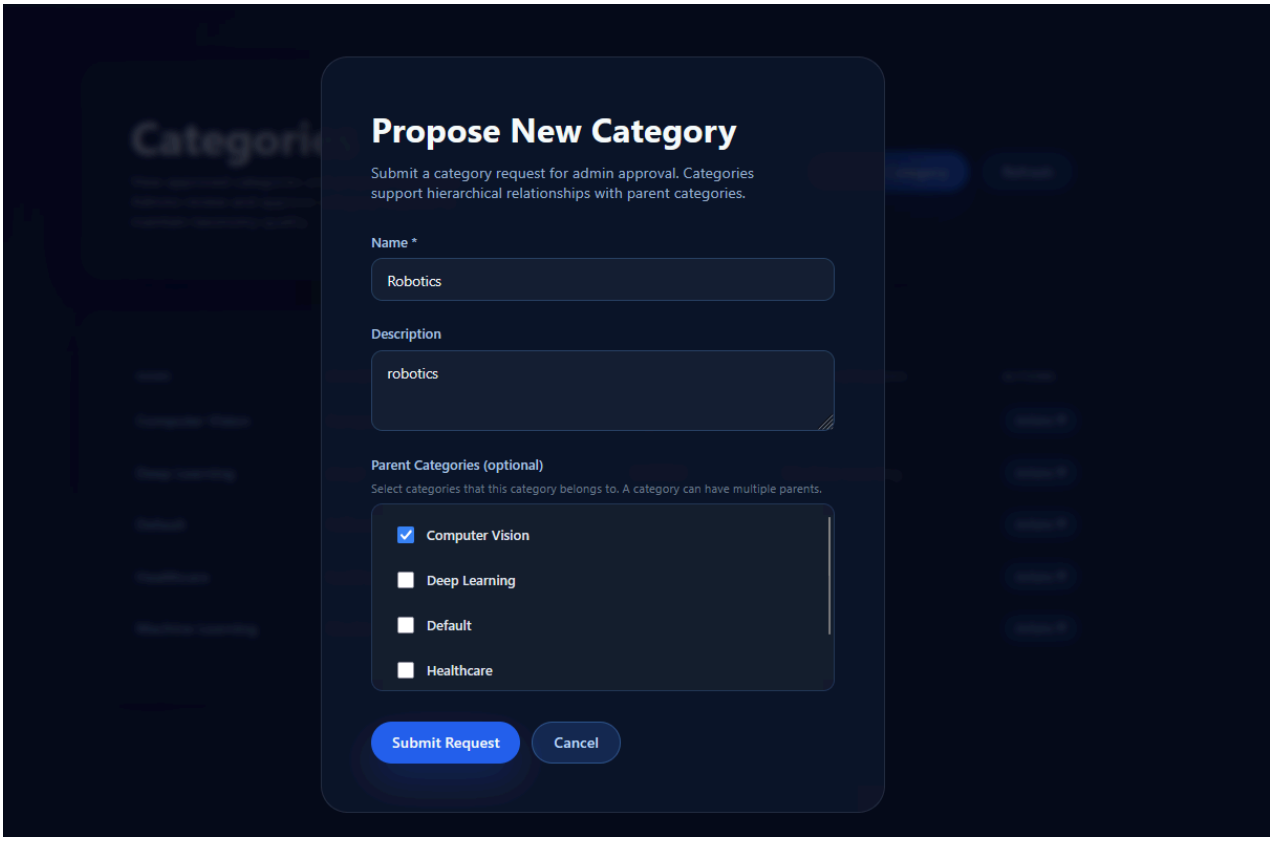
Εικόνα 5.39: Μενού Πλοήγησης - Επιλογή Κατηγοριών



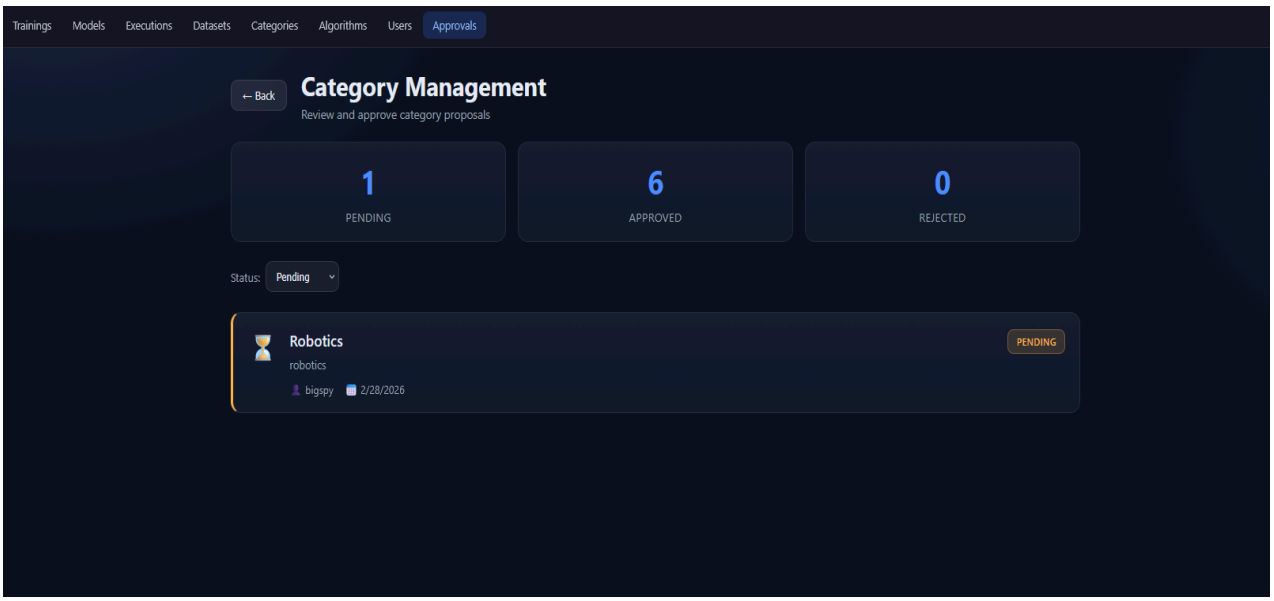
Εικόνα 5.40: Σελίδα Κατηγοριών



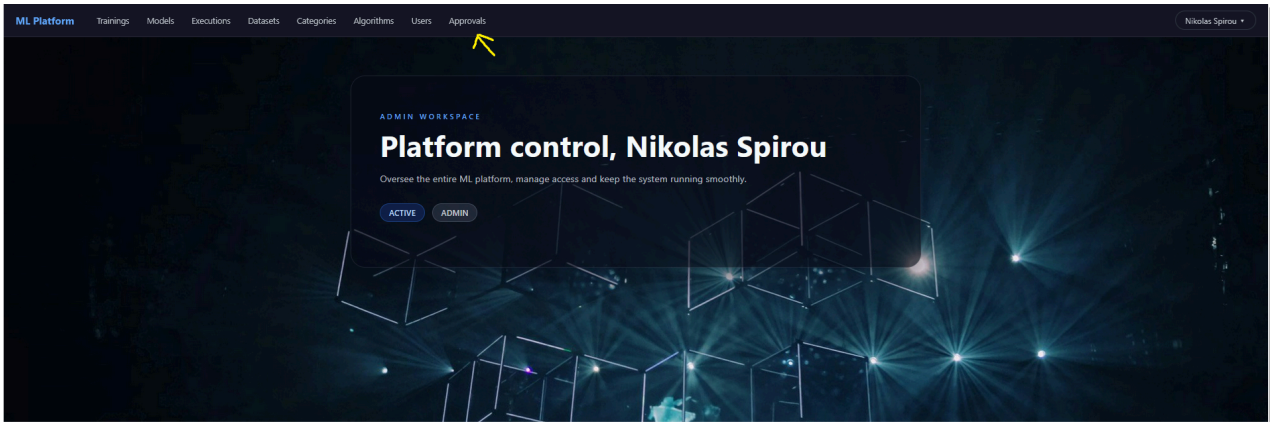
Εικόνα 5.41: Πλοήγηση στη Δημιουργία Κατηγορίας



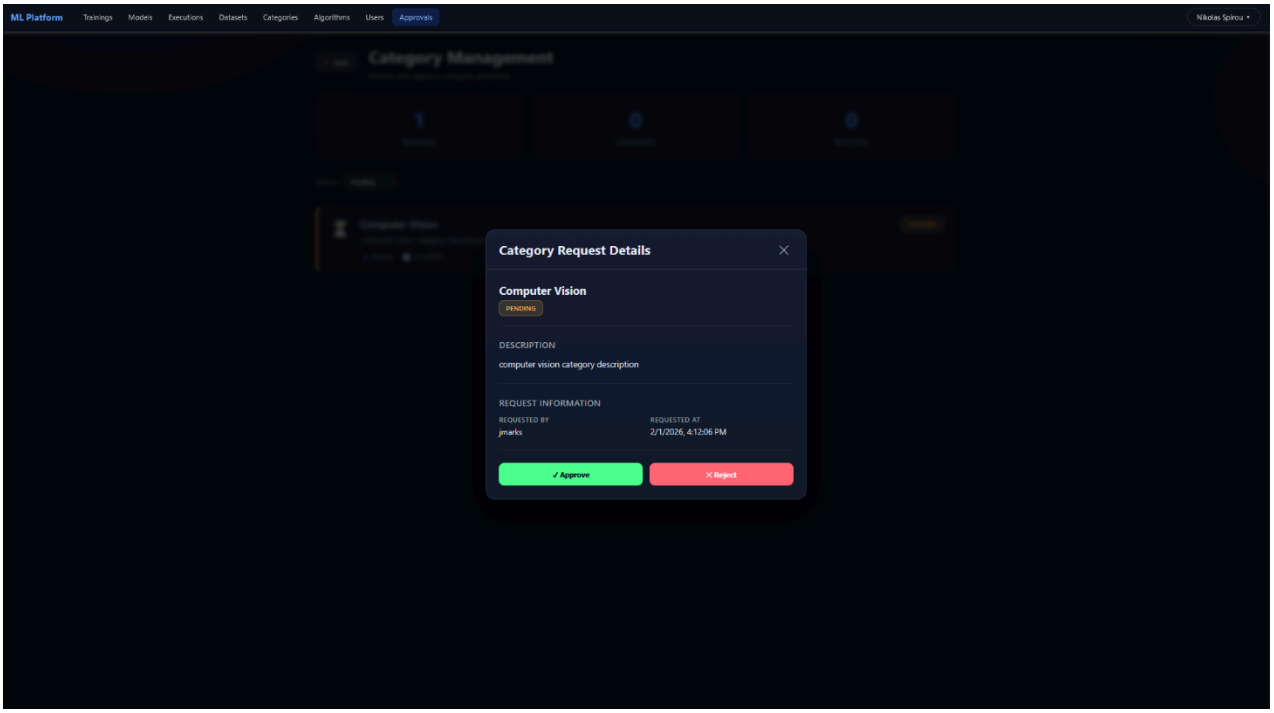
Εικόνα 5.42: Αίτημα Δημιουργίας Κατηγορίας (User)



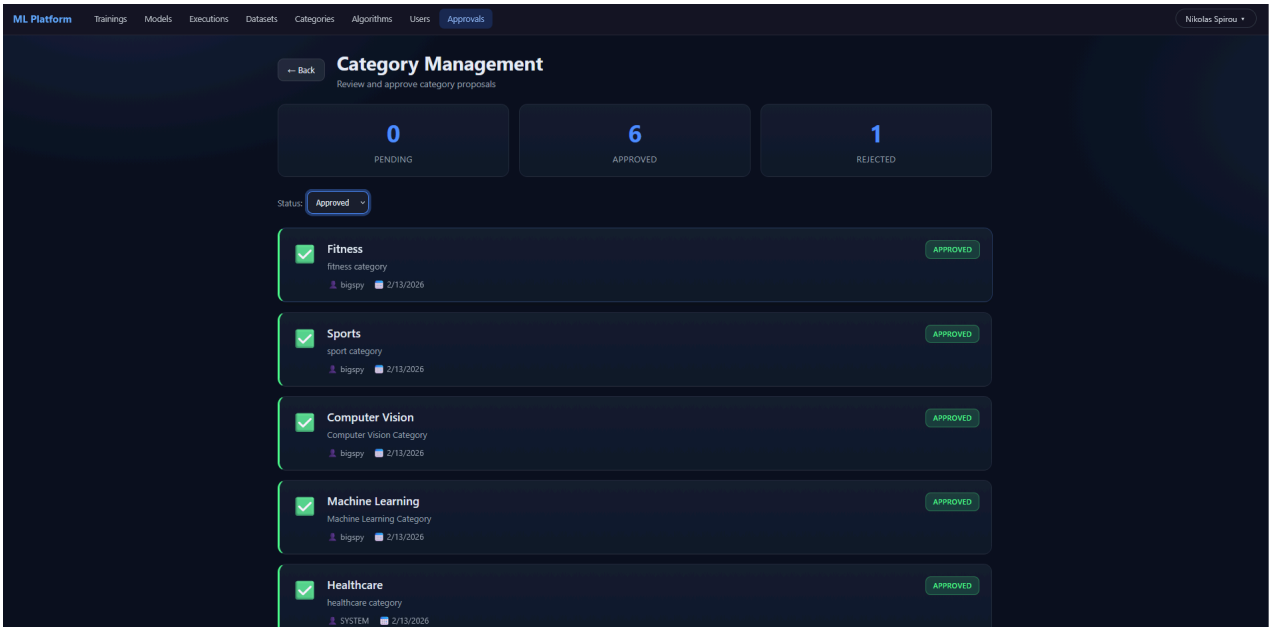
Εικόνα 5.43: Σελίδα Διαχείρισης Κατηγοριών (Pending)



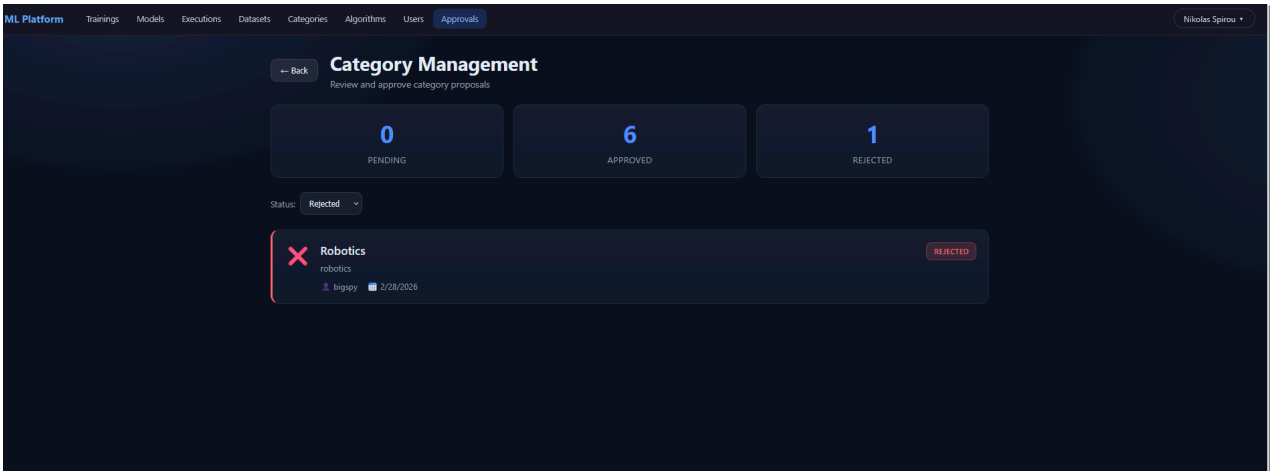
Εικόνα 5.44: Μενού Πλοήγησης - Επιλογή Διαχείρισης Κατηγοριών



Εικόνα 5.45: Αποδοχή/Απόρριψη Αιτήματος Δημιουργίας Κατηγορίας (Admin)



Εικόνα 5.46: Σελίδα Διαχείρισης Κατηγοριών (Approved)



Εικόνα 5.47: Σελίδα Διαχείρισης Κατηγοριών (Rejected)

5.2.7 Σενάριο 7: Επανεκπαίδευση Μοντέλων

Στο σενάριο αυτό παρουσιάζεται η διαδικασία επανεκπαίδευσης ενός μοντέλου, η οποία επιτρέπει στον χρήστη να αξιοποιήσει τη διαμόρφωση μιας προηγούμενης εκπαίδευσης (αλγόριθμο, παραμέτρους, στήλες) ως αφετηρία, τροποποιώντας επιλεκτικά στοιχεία όπως το σύνολο δεδομένων ή τις παραμέτρους του αλγορίθμου. Η λειτουργία αυτή διευκολύνει τον πειραματισμό και τη σταδιακή βελτίωση μοντέλων.

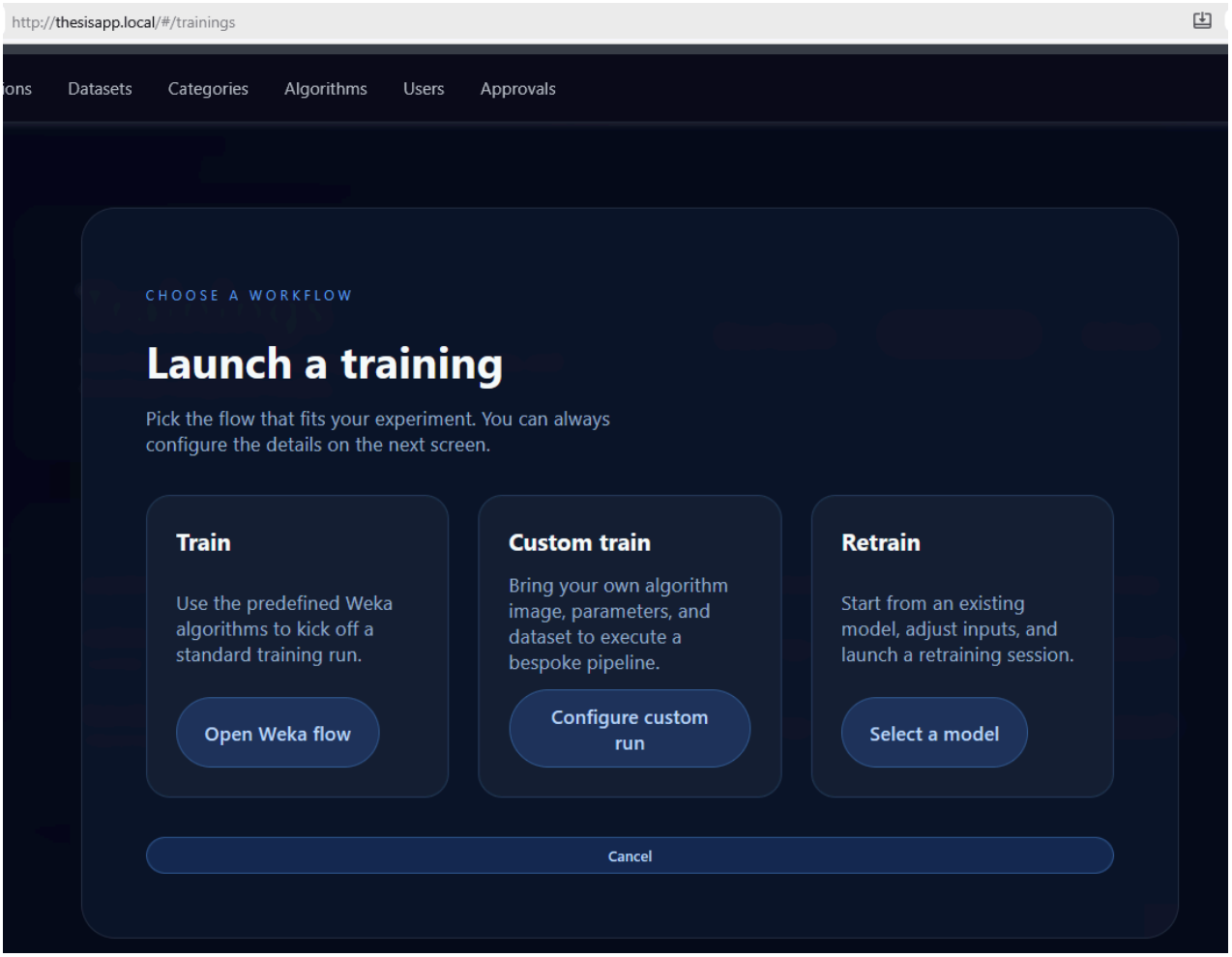
Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος και διαθέτει τουλάχιστον μία ολοκληρωμένη εκπαίδευση ή ένα εκπαιδευμένο μοντέλο (βλ. Σενάριο 1 ή 3).

Βήματα:

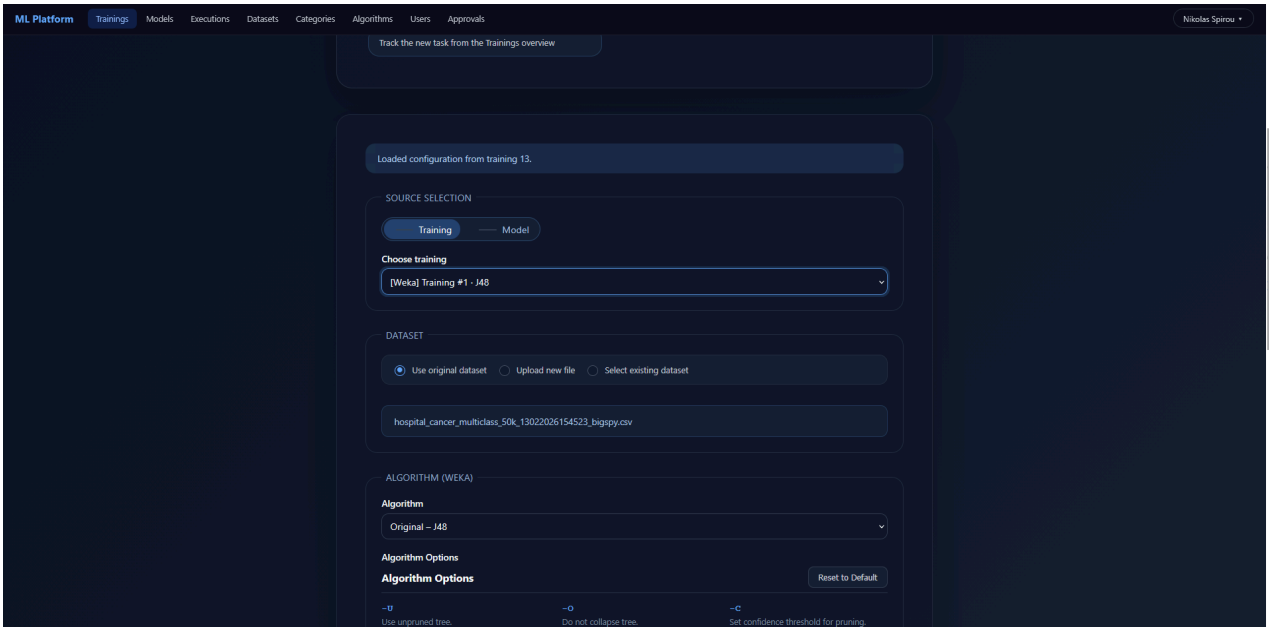
1. Ο χρήστης επιλέγει το στοιχείο Trainings από το κεντρικό μενού πλοήγησης. Εμφανίζεται η σελίδα διαχείρισης εκπαιδεύσεων.
2. Ο χρήστης πατά το κουμπί Start Training και από τον διάλογο επιλογής τύπου εκπαίδευσης (δείτε Εικόνα 5.48) επιλέγει Retrain, μεταβαίνοντας στη φόρμα επανεκπαίδευσης.
3. Στη φόρμα (δείτε Εικόνα 5.49), ο χρήστης επιλέγει την πηγή της επανεκπαίδευσης μέσω εναλλαγής (toggle) μεταξύ δύο επιλογών:
   - From Training: Επιλογή από τη λίστα προηγούμενων εκπαιδεύσεων.
   - From Model: Επιλογή από τη λίστα εκπαιδευμένων μοντέλων.

   Με την επιλογή, η φόρμα συμπληρώνεται αυτόματα με τη διαμόρφωση της αρχικής εκπαίδευσης (αλγόριθμος, παράμετροι, ρόλοι στηλών).

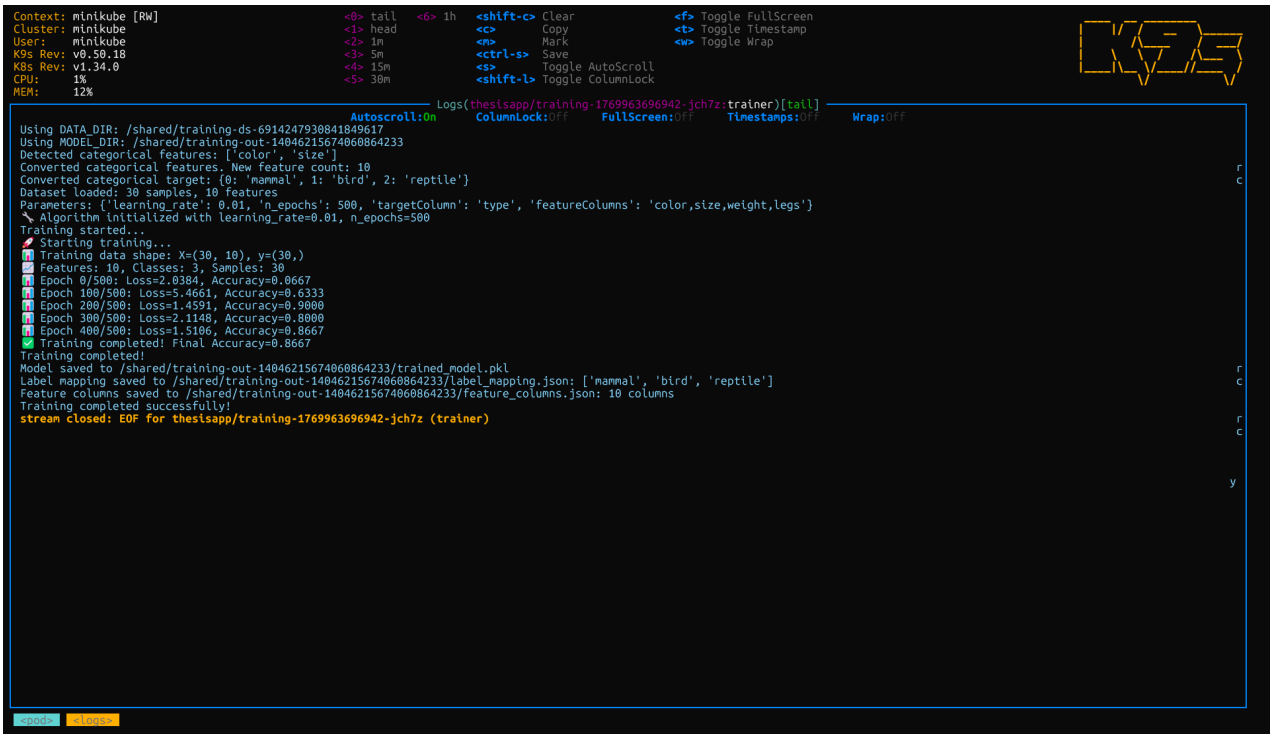
4. Ο χρήστης επιλέγει το σύνολο δεδομένων για την επανεκπαίδευση (δείτε Εικόνα 5.49), με τρεις διαθέσιμες επιλογές:
   - Χρήση του αρχικού συνόλου δεδομένων της προηγούμενης εκπαίδευσης.
   - Επιλογή υπάρχοντος συνόλου δεδομένων από τη λίστα μεταφορτωμένων συνόλων.
   - Μεταφόρτωση νέου συνόλου δεδομένων.
5. Ο χρήστης τροποποιεί, εφόσον επιθυμεί, τις παραμέτρους του αλγορίθμου και τη διαμόρφωση στηλών, και πατά το κουμπί Start Training για την εκκίνηση της επανεκπαίδευσης.
6. Η εκτέλεση ακολουθεί την ίδια ασύγχρονη διαδικασία με τα Σενάρια 1 και 3: δημιουργείται Kubernetes Job (δείτε Εικόνα 5.50), ο χρήστης παρακολουθεί την κατάσταση μέσω του πάνελ ενεργών εργασιών και με την ολοκλήρωση το νέο μοντέλο αποθηκεύεται στο σύστημα.



Εικόνα 5.48: Επιλογή Τύπου Εκπαίδευσης



Εικόνα 5.49: Επανεκπαίδευση Προσαρμοσμένου Αλγορίθμου



Εικόνα 5.50: Kubernetes Pod Επανεκπαίδευση

5.2.8 Σενάριο 8: Διακοπή Εκπαίδευσης/Πρόβλεψης

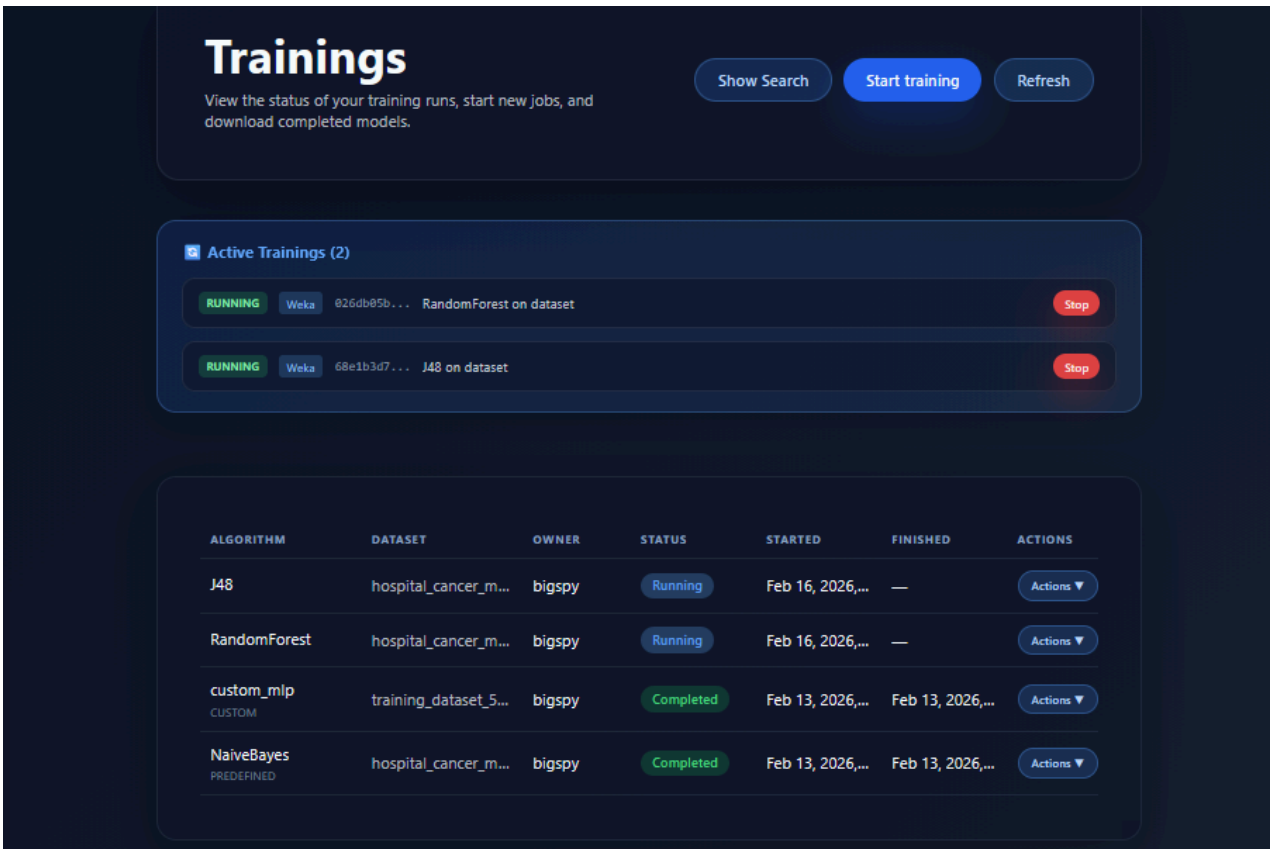
Στο σενάριο αυτό παρουσιάζεται η δυνατότητα ακύρωσης μιας εργασίας (εκπαίδευσης ή πρόβλεψης) που βρίσκεται σε εξέλιξη. Η λειτουργία αυτή είναι ιδιαίτερα χρήσιμη σε

περιπτώσεις εσφαλμένης παραμετροποίησης ή υπερβολικά χρονοβόρων εκτελέσεων, καθώς επιτρέπει την άμεση απελευθέρωση των δεσμευμένων υπολογιστικών πόρων του cluster.

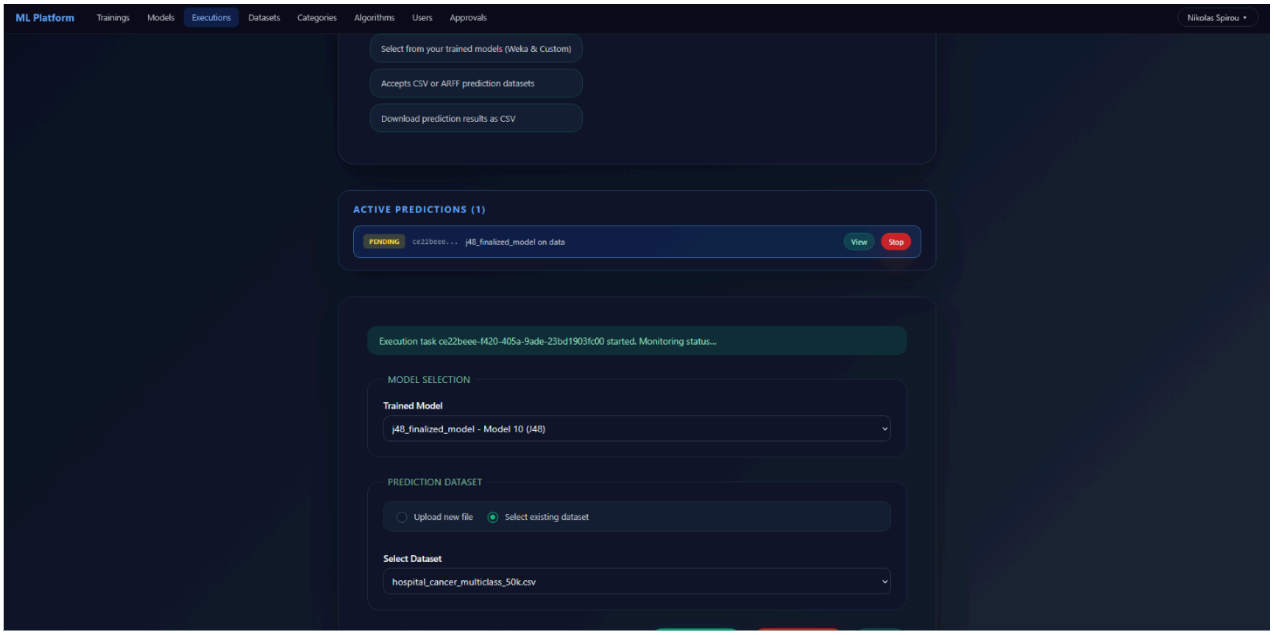
Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος και έχει τουλάχιστον μία εργασία σε κατάσταση PENDING ή RUNNING.

Βήματα:

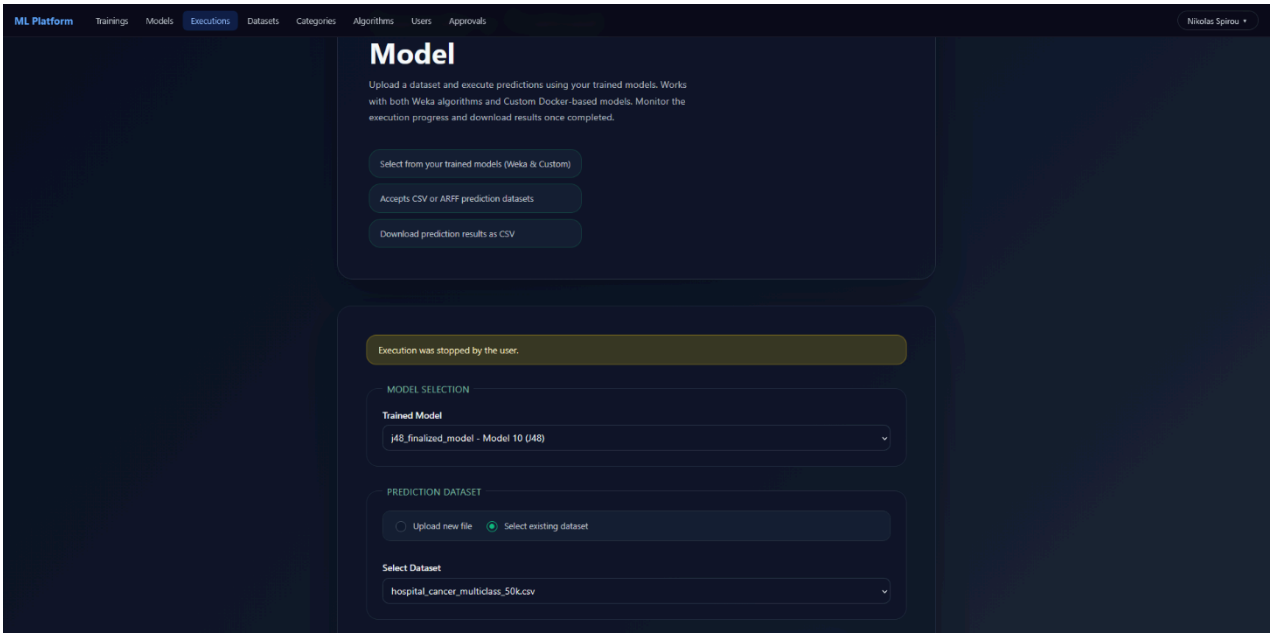
1. Κατά τη διάρκεια μιας εκπαίδευσης ή πρόβλεψης, ο χρήστης βλέπει την κατάσταση της εργασίας στο πάνελ ενεργών εργασιών (Active Tasks) που εμφανίζεται στη φόρμα εκπαίδευσης ή εκτέλεσης πρόβλεψης (δείτε Εικόνα 5.51).
2. Ο χρήστης πατά το κουμπί Stop για Training (δείτε Εικόνα 5.51) και Stop για Execution (δείτε Εικόνα 5.52), ανάλογα με τον τύπο της εργασίας.
3. Η εφαρμογή αποστέλλει αίτημα διακοπής στο backend, το οποίο τερματίζει το αντίστοιχο Kubernetes Job (δείτε Εικόνα 5.54) και απελευθερώνει τους δεσμευμένους πόρους (CPU, μνήμη) του cluster.
4. Ο χρήστης βλέπει το αντίστοιχο μήνυμα επιτυχής διακοπής (δείτε Εικόνα 5.53) και μπορεί να εκκινήσει νέα εργασία χωρίς να αναμένει την ολοκλήρωση της προηγούμενης.



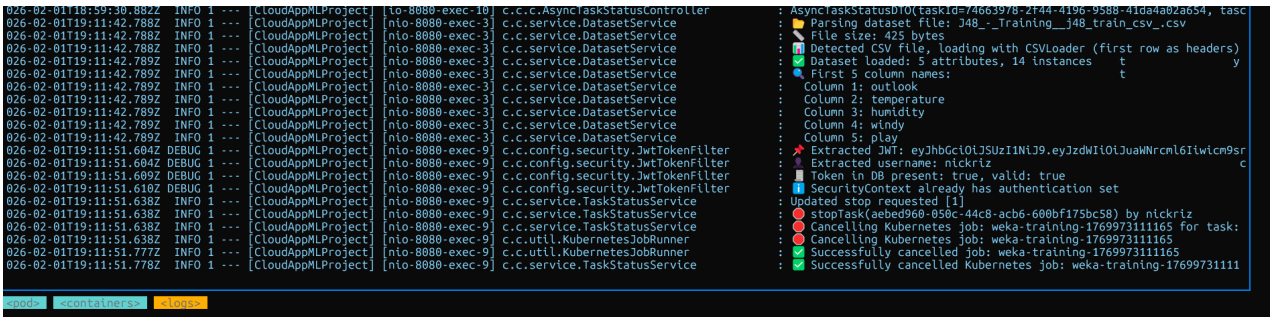
Εικόνα 5.51: Διακοπή Εκπαίδευσης



Εικόνα 5.52: Διακοπή Πρόβλεψης



Εικόνα 5.53 Μήνυμα Επιτυχίας Διακοπής Πρόβλεψης



Εικόνα 5.54: Διακοπή Εκπαίδευσης K8s Logs

5.2.9 Σενάριο 9: Μεταφόρτωση και Διαχείριση Δεδομένων

Στο σενάριο αυτό παρουσιάζεται η διαδικασία μεταφόρτωσης ενός νέου συνόλου δεδομένων στο σύστημα, καθώς και οι βασικές λειτουργίες διαχείρισής του. Η μεταφόρτωση αποτελεί προαπαιτούμενο βήμα για την εκπαίδευση μοντέλων και την εκτέλεση προβλέψεων.

Προϋποθέσεις: Ο χρήστης είναι αυθεντικοποιημένος στο σύστημα.

Βήματα:

1 Ο χρήστης επιλέγει το στοιχείο Datasets από το κεντρικό μενού πλοήγησης (δείτε Εικόνα 5.55). Εμφανίζεται η σελίδα διαχείρισης συνόλων δεδομένων (δείτε Εικόνα 5.57), η οποία περιλαμβάνει τη λίστα των υπαρχόντων συνόλων σε μορφή

σελιδοποιημένου πίνακα, με δυνατότητα φιλτραρίσματος κατά τύπο (Training, Prediction, All).

2 Ο χρήστης πατά το κουμπί Upload Dataset, οπότε εμφανίζεται ο διάλογος μεταφόρτωσης.

3 Στον διάλογο (δείτε Εικόνα 5.56), ο χρήστης:

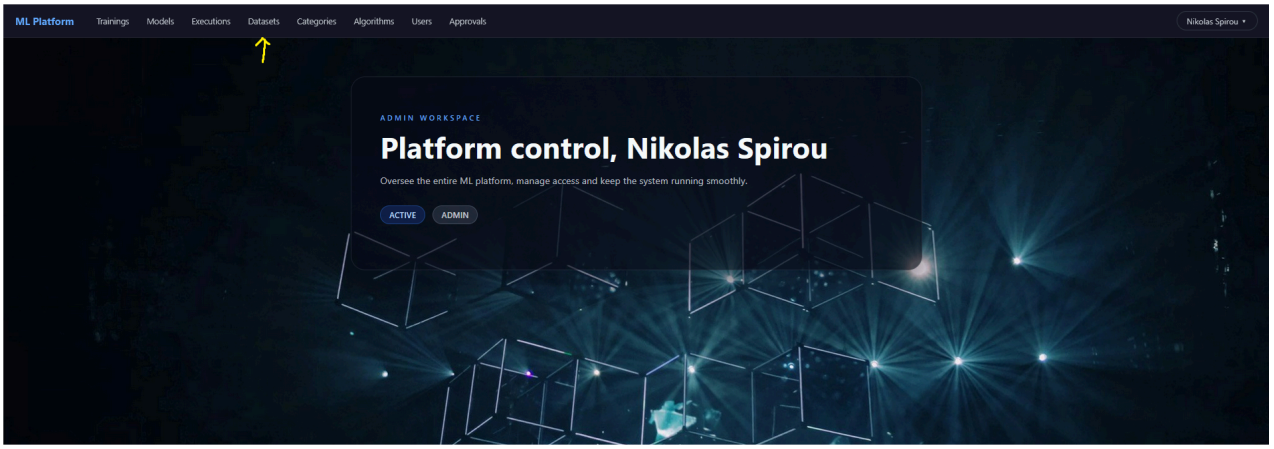
- Επιλέγει το αρχείο δεδομένων (υποστηριζόμενες μορφές: CSV, ARFF, XLSX) μέσω drag-and-drop ή επιλογής αρχείου.
- Εισάγει μια σύντομη περιγραφή του συνόλου δεδομένων.
- Καθορίζει το επίπεδο προσβασιμότητας: Private (ορατό μόνο στον ιδιοκτήτη), Public (ορατό σε όλους τους χρήστες) ή Shared (ορατό σε επιλεγμένους χρήστες).
- Επιλέγει τον λειτουργικό τύπο: Training (για εκπαίδευση μοντέλων) ή Prediction (για εκτέλεση προβλέψεων).

4 Με την υποβολή, το αρχείο μεταφορτώνεται στο αποθετήριο αντικειμένων και τα μεταδεδομένα καταχωρίζονται στη βάση δεδομένων. Το σύνολο δεδομένων εμφανίζεται πλέον στη λίστα (δείτε Εικόνα 5.57), όπου ο χρήστης μπορεί να δει πληροφορίες, όπως το όνομα αρχείου, το μέγεθος, τη μορφή, τον τύπο, τον ιδιοκτήτη, την ημερομηνία μεταφόρτωσης και τον αριθμό εκπαιδεύσεων που έχουν πραγματοποιηθεί με αυτό.

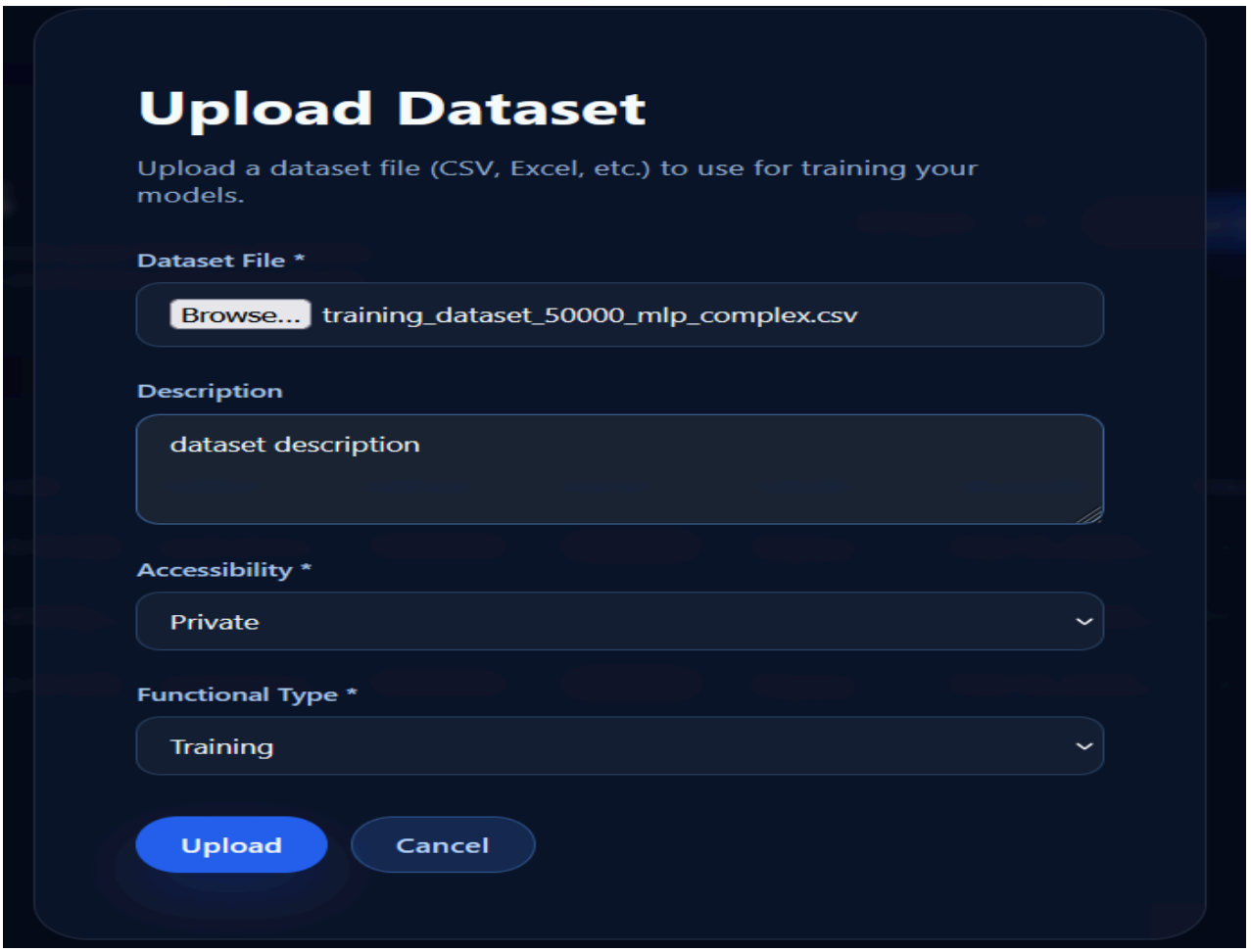
5 Από τη λίστα, ο χρήστης μπορεί να εκτελέσει επιπλέον ενέργειες μέσω του μενού επιλογών κάθε συνόλου δεδομένων:

- Download: Λήψη του αρχείου στον τοπικό υπολογιστή.
- Delete: Διαγραφή του συνόλου δεδομένων (διαθέσιμη μόνο στον ιδιοκτήτη).

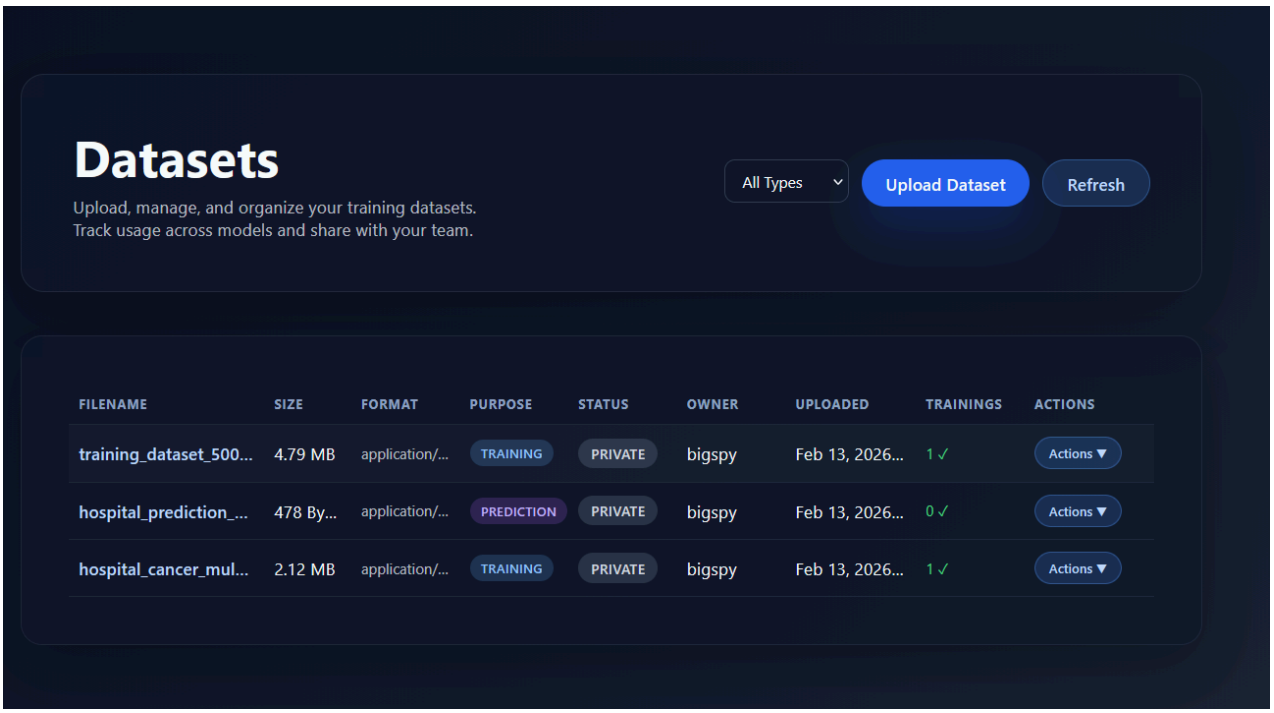
Τα σύνολα δεδομένων που έχουν οριστεί ως Public είναι ορατά και χρησιμοποιήσιμα από όλους τους χρήστες κατά τη δημιουργία εκπαιδεύσεων ή προβλέψεων, ενώ τα Private είναι προσβάσιμα μόνο από τον ιδιοκτήτη τους.



Εικόνα 5.55: Μενού Πλοήγησης - Διαχείριση Συνόλου Δεδομένων



Εικόνα 5.56: Μεταφόρτωση Συνόλου Δεδομένων



Εικόνα 5.57: Μεταφορτωμένα Σύνολα Δεδομένων

5.3 Αποτίμηση της Εφαρμογής

Η επίδειξη της εφαρμογής μας ThesisApp κατέδειξε ότι αυτή ανταποκρίνεται επιτυχώς στους βασικούς λειτουργικούς και αρχιτεκτονικούς στόχους που τέθηκαν κατά τον σχεδιασμό της. Η ασύγχρονη εκτέλεση εργασιών, η απομόνωση πόρων μέσω Kubernetes Jobs και η αξιοποίηση object storage για τη διαχείριση δεδομένων και αποτελεσμάτων συνέβαλαν στη σταθερή και προβλέψιμη συμπεριφορά του συστήματος.

Η αρχιτεκτονική της εφαρμογής αποδείχθηκε ικανή να υποστηρίξει τόσο προκαθορισμένους αλγορίθμους μηχανικής μάθησης όσο και προσαρμοσμένους αλγορίθμους που παρέχονται από τους χρήστες, χωρίς να απαιτείται τροποποίηση του πυρήνα του συστήματος. Επιπλέον, η δυνατότητα παρακολούθησης και ακύρωσης εργασιών ενισχύει την αξιοπιστία και την αποδοτική χρήση υπολογιστικών πόρων. Συνολικά, τα αποτελέσματα της επίδειξης επιβεβαιώνουν ότι η εφαρμογή αποτελεί μια λειτουργικά ολοκληρωμένη και αρχιτεκτονικά συνεπή πλατφόρμα μηχανικής μάθησης, κατάλληλη για εκτέλεση σε σύγχρονα περιβάλλοντα υπολογιστικού νέφους.

6 Συμπεράσματα και Μελλοντική Εργασία

Το παρόν κεφάλαιο συνοψίζει τα βασικά συμπεράσματα που προκύπτουν από την υλοποίηση και αξιολόγηση της εφαρμογής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας. Αρχικά, παρουσιάζεται συνοπτική αποτίμηση της συνεισφοράς και των πλεονεκτημάτων του προτεινόμενου συστήματος, στη συνέχεια καταγράφονται οι εμπειρίες που αποκομίστηκαν κατά τη διαδικασία υλοποίησης και, τέλος, προτείνονται κατευθύνσεις για μελλοντική εργασία και περαιτέρω εξέλιξη της εφαρμογής.

6.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία επικεντρώθηκε στον σχεδιασμό και την υλοποίηση μιας σύγχρονης εφαρμογής υπολογιστικού νέφους για τη δημιουργία, εκπαίδευση, διαχείριση και εκτέλεση μοντέλων μηχανικής μάθησης σε πολυχρηστικό περιβάλλον.

Η ανάπτυξη της εφαρμογής ακολούθησε επαναληπτικό μοντέλο (iterative development), όπου κάθε κύκλος ανάπτυξης (increment) εμπλούτιζε το σύστημα με νέες λειτουργίες και βελτιώσεις. Η προσέγγιση αυτή επέτρεψε τον σταδιακό επανασχεδιασμό της αρχιτεκτονικής και των βασικών μηχανισμών, με στόχο την επίτευξη ενός συστήματος επεκτάσιμου, ευέλικτου και τεχνικά βιώσιμου σε βάθος χρόνου. Κατά τη διάρκεια της υλοποίησης, δόθηκε ιδιαίτερη έμφαση στη γενίκευση και αφαίρεση της επιχειρησιακής λογικής, ώστε ο κώδικας να παραμένει όσο το δυνατόν πιο ανεξάρτητος από συγκεκριμένες βιβλιοθήκες μηχανικής μάθησης ή γλώσσες προγραμματισμού. Η προσέγγιση αυτή επέτρεψε τη δημιουργία ενός πλαισίου, το οποίο μπορεί να υποστηρίξει τόσο προκαθορισμένους αλγορίθμους από διαφορετικές βιβλιοθήκες όσο και προσαρμοσμένους αλγορίθμους που υλοποιούνται σε ετερογενή περιβάλλοντα.

Η υλοποιημένη αρχιτεκτονική βασίζεται σε σαφή διαχωρισμό ευθυνών και σε μηχανισμούς ασύγχρονης εκτέλεσης, επιτρέποντας την αποδοτική διαχείριση υπολογιστικά απαιτητικών διεργασιών. Παρά το γεγονός ότι το σύστημα υλοποιήθηκε ως μονολιθική εφαρμογή, η εσωτερική του δομή και ο λογικός διαχωρισμός των λειτουργιών επιτρέπουν τη μελλοντική μετάβαση σε αρχιτεκτονική μικροϋπηρεσιών (microservices), χωρίς εκτεταμένες αλλαγές στον βασικό σχεδιασμό του.

Ιδιαίτερα σημαντική συνεισφορά της εργασίας αποτελεί η ασφαλής ενσωμάτωση προσαρμοσμένων αλγορίθμων μέσω δοχειοποιημένης εκτέλεσης. Η επιλογή αυτή εξασφαλίζει απομόνωση, οριοθέτηση εμπιστοσύνης και ανεξαρτησία από την τεχνολογία υλοποίησης του αλγορίθμου, καθιστώντας την εφαρμογή κατάλληλη για πειραματική αλλά και παραγωγική χρήση. Παράλληλα, κατά την υλοποίηση αναδείχθηκαν ορισμένα σημεία που ικανοποιούνται μερικώς και αποτελούν αντικείμενο μελλοντικής βελτίωσης. Η παρακολούθηση προόδου εκπαιδεύσεων και εκτελέσεων βασίζεται σε μηχανισμό polling αντί για push-based ενημέρωση σε πραγματικό χρόνο, επιλογή που μειώνει την αρχιτεκτονική πολυπλοκότητα αλλά περιορίζει την αμεσότητα της ενημέρωσης. Το REST API υλοποιεί επίπεδο ωριμότητας 2 χωρίς υποστήριξη HATEOAS, ενώ ορισμένοι μηχανισμοί ασφαλείας (TLS σε επίπεδο εφαρμογής, προστασία XSS/SSRF) προβλέπονται για παραγωγικό περιβάλλον. Η υποστήριξη ταυτόχρονων χρηστών δεν έχει επαληθευτεί μέσω δοκιμών φόρτου, και η responsive διεπαφή απαιτεί περαιτέρω βελτιστοποίηση σε σύνθετες οθόνες μικρών συσκευών. Τα σημεία αυτά δεν υποδηλώνουν δομική αδυναμία, αλλά αντανακλούν τη συνειδητή προτεραιοποίηση της αρχιτεκτονικής καθαρότητας και της υποστήριξης των βασικών σεναρίων μηχανικής μάθησης.Συνολικά, η εργασία καταδεικνύει ότι είναι εφικτή η ανάπτυξη μιας επεκτάσιμης και τεχνικά ώριμης πλατφόρμας μηχανικής μάθησης σε περιβάλλοντα νέφους.

6.2 Εμπειρίες από την Υλοποίηση

Η διαδικασία ανάπτυξης της εφαρμογής ανέδειξε τη σημασία του προσεκτικού αρχιτεκτονικού σχεδιασμού σε συστήματα μηχανικής μάθησης μεγάλης κλίμακας. Η ανάγκη υποστήριξης ασύγχρονων και μακρόχρονων διεργασιών κατέστησε απαραίτητη την υιοθέτηση μηχανισμών παρακολούθησης κατάστασης, διαχείρισης αποτυχιών και ασφαλούς διακοπής εκτελέσεων.

Παράλληλα, η υλοποίηση γενικών και επεκτάσιμων μηχανισμών για την εκπαίδευση και εκτέλεση μοντέλων ανέδειξε τη δυσκολία εξισορρόπησης μεταξύ ευελιξίας και πολυπλοκότητας. Η εμπειρία αυτή κατέδειξε ότι η επένδυση σε αφαιρετικά και γενικευμένα σχήματα σχεδίασης, παρότι αυξάνει το αρχικό κόστος υλοποίησης, αποδίδει μακροπρόθεσμα σε συντηρησιμότητα και δυνατότητα επέκτασης.

6.3 Περιορισμοί και Σημεία Βελτίωσης

Παρά τη λειτουργική πληρότητα του συστήματος, η τρέχουσα υλοποίηση παρουσιάζει ορισμένους περιορισμούς που αναγνωρίστηκαν κατά τη διαδικασία ανάπτυξης και αξιολόγησης:

- *Σύστημα ρόλων*: Το σύστημα υποστηρίζει δύο βασικούς ρόλους (Admin, User). Η προσθήκη εξειδικευμένων ρόλων (π.χ. Data Scientist, ML Engineer, Project Manager) με διαφοροποιημένα δικαιώματα ανά λειτουργία θα ενίσχυε τη χρηστικότητα σε ομαδικά περιβάλλοντα εργασίας.
- *Διεπαφή χρήστη*: Η τρέχουσα διεπαφή καλύπτει τις βασικές λειτουργίες, ωστόσο θα μπορούσε να βελτιωθεί ως προς την οπτικοποίηση αποτελεσμάτων, τη σύγκριση μοντέλων μεταξύ τους και τη συνολική εμπειρία χρήστη.
- *Υποστηριζόμενοι αλγόριθμοι*: Οι προκαθορισμένοι αλγόριθμοι βασίζονται αποκλειστικά στη βιβλιοθήκη Weka. Η ενσωμάτωση αλγορίθμων από επιπλέον βιβλιοθήκες (π.χ. scikit-learn, XGBoost) καθώς και αλγορίθμων βαθιάς μάθησης (deep learning) μέσω πλαισίων όπως TensorFlow ή PyTorch θα διεύρυνε σημαντικά το φάσμα εφαρμογής της πλατφόρμας.

6.4 Μελλοντική Εργασία και Όραμα Εξέλιξης

Παρότι η εφαρμογή καλύπτει τους βασικούς στόχους της παρούσας διπλωματικής, προσφέρονται σημαντικές δυνατότητες περαιτέρω εξέλιξής της.

Σε επίπεδο αρχιτεκτονικής, η εφαρμογή μπορεί να εξελιχθεί σε κατανεμημένη πλατφόρμα βασισμένη σε μικροϋπηρεσίες, με ανεξάρτητη κλιμάκωση επιμέρους λειτουργιών — όπως η διαχείριση δεδομένων, η εκπαίδευση μοντέλων και η εκτέλεση προβλέψεων — ώστε κάθε υποσύστημα να κλιμακώνεται αυτόνομα ανάλογα με τον φόρτο εργασίας. Η ενσωμάτωση μηχανισμών αυτοματοποιημένης κλιμάκωσης θα βελτίωνε σημαντικά την αξιοποίηση πόρων: σε επίπεδο pods μέσω Horizontal Pod Autoscaler (HPA) με αξιοποίηση custom metrics (π.χ. αριθμός εκκρεμών εργασιών εκπαίδευσης ή πρόβλεψης), και σε επίπεδο υποδομής μέσω Cluster Autoscaler για δυναμική προσθήκη κόμβων όταν οι υπάρχοντες πόροι δεν επαρκούν.

Σε επίπεδο υλικού, μία ουσιώδης επέκταση αφορά την ενσωμάτωση υποστήριξης GPU μέσω του NVIDIA Container Runtime, ώστε οι εργασίες εκπαίδευσης να αξιοποιούν επιτάχυνση GPU εντός των containers. Παράλληλα, η πλατφόρμα θα μπορούσε να προσφέρει στους χρήστες τη δυνατότητα επιλογής υπολογιστικών μηχανημάτων με διαφορετικά χαρακτηριστικά υλικού (υπολογιστική ισχύς, μνήμη, αποθηκευτικός χώρος), προσαρμόζοντας τους διαθέσιμους πόρους στις απαιτήσεις κάθε εργασίας.

Μία βασική κατεύθυνση αφορά την υποστήριξη κατανεμημένης εκπαίδευσης μοντέλων σε υβριδικά περιβάλλοντα (hybrid cloud, fog/edge computing), αξιοποιώντας πλαίσια όπως το Apache Spark για παράλληλη επεξεργασία σε πολλαπλούς κόμβους. Η προσέγγιση αυτή θα μείωνε τον χρόνο εκπαίδευσης και θα επέτρεπε τη χρήση μοντέλων σε περιβάλλοντα με αυστηρούς περιορισμούς καθυστέρησης ή ιδιωτικότητας δεδομένων. Επιπλέον, θα μπορούσε να προστεθεί υποστήριξη παύσης (pause) και επανεκκίνησης (resume) διεργασιών εκπαίδευσης, επιτρέποντας πιο ευέλικτη διαχείριση υπολογιστικών πόρων και καλύτερη αξιοποίηση του χρόνου εκτέλεσης.

Στον τομέα της διαχείρισης μοντέλων, η υλοποίηση προηγμένου συστήματος εκδοσιοποίησης (model versioning) θα επέτρεπε τη συστηματική παρακολούθηση της εξέλιξης των μοντέλων και τη διενέργεια δοκιμών A/B testing μεταξύ διαφορετικών εκδόσεων, διευκολύνοντας την αξιολόγηση και την ελεγχόμενη μετάβαση σε νεότερα μοντέλα.

Μία ακόμη σημαντική κατεύθυνση αφορά τη δημιουργία σύνθετων, παραμετροποιήσιμων ροών εργασίας (custom pipelines). Η τρέχουσα υλοποίηση παρέχει προκαθορισμένες ροές εκπαίδευσης και πρόβλεψης. Σε μελλοντική έκδοση, οι χρήστες θα μπορούσαν να σχεδιάζουν δικές τους ροές που θα περιλαμβάνουν στάδια προεπεξεργασίας δεδομένων (καθαρισμός, μετασχηματισμός, κανονικοποίηση), εκπαίδευσης και αξιολόγησης, συνθέτοντας διαδοχικά βήματα τα οποία θα υποστηρίζονται είτε από ενσωματωμένες λειτουργίες της πλατφόρμας είτε από προσαρμοσμένα βήματα μέσω containers που παρέχει ο χρήστης. Παράλληλα, η υποστήριξη τόσο μαζικής (batch) όσο και πραγματικού χρόνου (streaming) επεξεργασίας δεδομένων θα επέτρεπε τη δυναμική ενημέρωση μοντέλων καθώς εισέρχονται νέα δεδομένα.

Σε επίπεδο δεδομένων, η υποστήριξη διαχείρισης δεδομένων μεγάλης κλίμακας (big data) μέσω ενσωμάτωσης κατανεμημένης επεξεργασίας αποτελεί απαραίτητη προϋπόθεση για την κλιμάκωση της πλατφόρμας σε παραγωγικά περιβάλλοντα. Επιπλέον, η μετάβαση σε μια αρχιτεκτονική πλήρως προσανατολισμένη στην αποθήκευση στο cloud (cloud-based metadata storage) — όχι μόνο για τα αρχεία εκπαίδευσης και

πρόβλεψης αλλά και για τα μεταδεδομένα και τις ρυθμίσεις της πλατφόρμας — θα ενίσχυε τη φορητότητα και την ανθεκτικότητα του συστήματος.

Όσον αφορά τη βιβλιοθήκη μηχανικής μάθησης, η αναβάθμιση της Weka στην τελευταία διαθέσιμη έκδοση θα επέτρεπε την αξιοποίηση νέων αλγορίθμων και βελτιωμένων δυνατοτήτων παραμετροποίησης, διατηρώντας την πλατφόρμα ενημερωμένη με τις τρέχουσες εξελίξεις στον τομέα.

Τέλος, ένα μακροπρόθεσμο όραμα της πλατφόρμας αφορά τη μετεξέλιξή της σε ένα ευρύτερο οικοσύστημα τεχνητής νοημοσύνης, το οποίο θα υποστηρίζει τη δημιουργία και τον συντονισμό ευφυών πρακτόρων (agents), καθώς και τη σύνθεση σύνθετων συστημάτων τεχνητής νοημοσύνης βασισμένων σε συνεργαζόμενες οντότητες. Η κατεύθυνση αυτή καθιστά την παρούσα εργασία ένα ισχυρό θεμέλιο για μελλοντική ερευνητική και αναπτυξιακή δραστηριότητα στον χώρο των cloud-native συστημάτων τεχνητής νοημοσύνης.

Βιβλιογραφία

- Hastie, T. T. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.*
- Jordan, M. I. (2015). *Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255–260.*
- Ricci, F. R. (2015). *Recommender Systems Handbook. Springer.*
- Sutton, R. S. (2018). *Reinforcement Learning: An Introduction (2nd ed.). MIT Press.*
- M. Ribeiro, K. Grolinger and M. A. M. Capretz, "MLaaS: Machine Learning as a Service," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, 2015, pp. 896-902, doi: 10.1109/ICMLA.2015.152.
- Robert Philipp, Andreas Mladenow, Christine Strauss, and Alexander Völz. 2021. Machine Learning as a Service: Challenges in Research and Applications. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20)*. Association for Computing Machinery, New York, NY, USA, 396–406. https://doi.org/10.1145/3428757.3429152
- AzureML Team: Anatomy of a machine learning service, Microsoft - 1 Memorial Drive, Cambridge, MA 02142AzureML: Anatomy of a machine learning service
- https://ml.cms.waikato.ac.nz/weka/Witten_et_al_2016_appendix.pdf
- https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf
- https://kubernetes.io/docs/home/
- https://docs.docker.com/