



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Πολυτεχνική Σχολή

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΑΡΑΛΛΗΛΟ ΚΑΙ ΚΑΤΑΝΕΜΗΜΕΝΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ  
ΗΛΕΚΤΡΟΝΙΚΟΥ ΦΑΚΕΛΟΥ ΥΓΕΙΑΣ

Διπλωματική Εργασία

**ΒΟΥΖΑΣ ΑΝΤΩΝΙΟΣ**

Επιβλέπων καθηγητής  
Βασιλακόπουλος Μιχαήλ

Βόλος, Οκτώβριος 2023





UNIVERSITY OF THESSALY  
School of Engineering  
Department of Electrical and Computer Engineering

PARALLEL AND DISTRIBUTED ELECTRONIC HEALTH RECORD  
MANAGEMENT SYSTEM

Diploma Thesis

**VOUZAS ANTONIOS**

**Supervisor**  
Michael Vassilakopoulos

Volos, October 2023

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων

**Βασιλακόπουλος Μιχαήλ**

Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος

**Τουσίδου Ελένη**

Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος

**Τσαλαπάτα Χαρίκλεια**

Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

## **ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ**

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελούν αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλουν οποιασδήποτε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχουν έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η Δηλών/ούσα

Ονοματεπώνυμο Φοιτητή/ήτριας

# **ΕΥΧΑΡΙΣΤΙΕΣ**

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλλαν και βοήθησαν στην πραγματοποίηση αυτής της διπλωματικής εργασίας. Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Βασιλακόπουλο Μιχαήλ, που μου έδωσε τη δυνατότητα να ασχοληθούμε με ένα τόσο ενδιαφέρον και σύγχρονο θέμα, καθώς επίσης και για την πολύτιμη καθοδήγησή και υποστήριξή του. Επιπλέον, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη συνεχή συμπαράσταση και υπομονή τους καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερα, θα ήθελα να ευχαριστήσω την την αδερφή μου που χωρίς την υποστήριξη και βοήθεια της, η εργασία αυτή δεν θα ήταν τόσο ολοκληρωμένη.

# ΠΕΡΙΛΗΨΗ

## Διπλωματική Εργασία

### ΠΑΡΑΛΛΗΛΟ ΚΑΙ ΚΑΤΑΝΕΜΗΜΕΝΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΦΑΚΕΛΟΥ ΥΓΕΙΑΣ

#### ΒΟΥΖΑΣ ΑΝΤΩΝΗΣ

Στην εποχή της τεχνολογικής εξέλιξης, τα συστήματα διαχείρισης Ηλεκτρονικού Φακέλου Υγείας (ΗΦΥ) αποτελούν αναπόσπαστο μέρος της σύγχρονης υγειονομικής περίθαλψης. Τα συστήματα αυτά επιτρέπουν την αποθήκευση, επεξεργασία και διαχείριση ιατρικών δεδομένων, γεγονός που μπορεί να οδηγήσει σε βελτιωμένη ποιότητα, αποτελεσματικότητα και ασφάλεια της παροχής υγειονομικής περίθαλψης. Ωστόσο, πολλά από τα υφιστάμενα συστήματα ΗΦΥ αντιμετωπίζουν διάφορες προκλήσεις, όπως η αποκεντρωμένη δομή, η έλλειψη συνοχής και η αδυναμία ανταλλαγής πληροφοριών μεταξύ τους. Στόχος της παρούσας διπλωματικής εργασίας είναι η αντιμετώπιση τέτοιων προκλήσεων μέσω της σχεδίασης και υλοποίησης ενός ολοκληρωμένου, ευέλικτου, ασφαλούς και αξιόπιστου συστήματος διαχείρισης ΗΦΥ. Η εργασία επικεντρώνεται στην ανάπτυξη μιας αρχιτεκτονικής που υποστηρίζει την παράλληλη και κατανεμημένη διαχείριση ιατρικών δεδομένων, αξιοποιώντας τις τεχνολογίες Kubernetes, Apache Cassandra, Apache Spark, FastAPI και ReactJS. Το αποτέλεσμα της εργασίας αυτής δεν αποτελεί απλώς μικρογραφία μιας εφαρμογής, αλλά επικεντρώνεται στην ανάπτυξη μιας γενικότερης αρχιτεκτονικής ενός συστήματος διαχείρισης ΗΦΥ.

Λέξεις Κλειδιά: Ηλεκτρονικός Φάκελος Υγείας, παράλληλο σύστημα, κατανεμημένο σύστημα, Kubernetes, Apache Cassandra, Apache Spark, FastAPI, ReactJS.

# **ABSTRACT**

Diploma Thesis

## **PARALLEL AND DISTRIBUTED ELECTRONIC HEALTH RECORD MANAGEMENT SYSTEM**

**VOUZAS ANTONIS**

In the era of technological advancement, Electronic Health Record (EHR) management systems have become an integral part of modern healthcare. These systems enable the storage, processing, and management of medical data, which can lead to improved quality, efficiency, and safety in healthcare delivery. However, many existing EHR systems face various challenges, such as decentralized structures, lack of cohesion, and inability to exchange information among them. The aim of this thesis is to address such challenges through the design and implementation of an integrated, flexible, secure, and reliable EHR management system. The work focuses on developing an architecture that supports parallel and distributed management of medical data, leveraging technologies like Kubernetes, Apache Cassandra, Apache Spark, FastAPI, and ReactJS. The outcome of this work is not merely a prototype application but aims at developing a broader architecture for an EHR management system.

**Keywords:** Electronic Health Record, parallel system, distributed system, Kubernetes, Apache Cassandra, Apache Spark, FastAPI, ReactJS.

# Περιεχόμενα

<b>ΕΥΧΑΡΙΣΤΙΕΣ</b>	iii
<b>ΠΕΡΙΛΗΨΗ</b>	iv
<b>ABSTRACT</b>	v
<b>ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ</b>	vi
<b>ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ</b>	viii
<b>1 ΕΙΣΑΓΩΓΗ</b>	1
1.1 Ορισμός των Προβλήματος . . . . .	1
1.2 Αντικείμενο εργασίας . . . . .	2
1.3 Συνεισφορά της εργασίας . . . . .	3
1.4 Δομή και μεθοδολογία εργασίας . . . . .	3
<b>2 ΗΛΕΚΤΡΟΝΙΚΟΣ ΦΑΚΕΛΟΣ ΥΓΕΙΑΣ</b>	5
2.1 Η εμφάνιση του ΗΦΥ . . . . .	5
2.2 Ορισμός και χαρακτηριστικά ΗΦΥ . . . . .	5
2.3 Ηλεκτρονικός Φάκελος Υγείας/Ηλεκτρονικός Ιατρικός Φάκελος . . . . .	7
2.4 Πλεονεκτήματα Ηλεκτρονικού Φακέλου Υγείας . . . . .	9
2.5 Ηλεκτρονικός Φάκελος Υγείας στην Ελλάδα . . . . .	10
2.6 Παραδείγματα συστημάτων ΗΦΥ . . . . .	15
2.6.1 OpenEMR . . . . .	15
2.6.2 GNU Health . . . . .	17
2.6.3 OpenMRS . . . . .	21
<b>3 ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ</b>	23
3.1 Kubernetes . . . . .	23
3.1.1 Helm Charts στο Kubernetes . . . . .	27
3.2 Apache Cassandra . . . . .	28
3.3 Apache Spark . . . . .	32

3.4	React JS . . . . .	35
3.5	FastAPI . . . . .	37
3.6	Nginx . . . . .	39
<b>4</b>	<b>ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΗΦΥ</b>	<b>41</b>
4.1	Στόχοι και Λειτουργίες του Συστήματος με βάση την Αρχιτεκτονική . . . . .	41
4.2	Σενάρια χρήσης . . . . .	43
4.3	Βάση δεδομένων Apache Cassandra . . . . .	45
4.3.1	Επιλογή βάσης δεδομένων . . . . .	45
4.3.2	Σχεδιασμός βάσης δεδομένων . . . . .	45
4.4	Ανάλυση Διεπαφής FastAPI . . . . .	53
4.4.1	Ανάλυση σημείων πρόσβασης (endpoints) . . . . .	53
4.4.2	Ανάλυση χαρακτηριστικών των σημείων πρόσβασης . . . . .	56
4.5	Ανάπτυξη Frontend με ReactJS . . . . .	62
4.5.1	Εφαρμογή για τον ασθενή . . . . .	62
4.5.2	Εφαρμογή για τον επαγγελματία υγείας . . . . .	66
4.5.3	Ανάλυση αρχιτεκτονικής και σχεδιασμού των εφαρμογών . . . . .	76
4.6	Ενσωμάτωση σε Kubernetes . . . . .	79
4.6.1	Ανάλυση αρχιτεκτονικής για ενσωμάτωση . . . . .	79
4.6.2	Περιγραφή και δημιουργία του cluster . . . . .	80
4.6.3	Ενσωμάτωση και Διαμόρφωση της Cassandra . . . . .	82
4.6.4	Ενσωματωση και Διαμόρφωση των εφαρμογών (Backend-Frontend) . . . . .	87
4.7	Ανάλυση με Spark . . . . .	90
4.7.1	Ενσωματωση και Διαμόρφωση Spark . . . . .	90
4.7.2	Ανάλυση δεδομένων με Spark . . . . .	92
<b>5</b>	<b>ΣΥΜΠΕΡΑΣΜΑΤΑ</b>	<b>95</b>
5.1	Σύνοψη και ανάλυση βασικών σημείων . . . . .	95
5.2	Περιορισμοί έρευνας . . . . .	96
5.3	Μελλοντικές Επεκτάσεις . . . . .	96
<b>ΒΙΒΛΙΟΓΡΑΦΙΑ</b>		<b>98</b>
<b>Παράρτημα Α</b>		<b>101</b>

# Κατάλογος Εικόνων

2.1	Κεντρική σελίδα (login) της εφαρμογής της Πρωτοβάθμιας Φροντίδας Υγείας [16]	11
2.2	Εισαγωγή AMKA [16]	11
2.3	Αρχική οθόνη εφαρμογής [16]	12
2.4	Επιλογή ΗΦΥ [16]	12
2.5	ΗΦΥ Ασθενούς-Επισκέψεις [16]	13
2.6	Ιστορικό Πρόσβασης [16]	13
2.7	Στοιχεία Προσώπου [16]	14
2.8	Διαγνώσεις [16]	14
2.9	Φαρμακευτική αγωγή [16]	14
2.10	Ενότητες [16]	15
2.11	Λογότυπο OpenEMR [17]	15
2.12	Αρχική Οθόνη και Είσοδος στο σύστημα [18]	16
2.13	Καρτέλα καταγραφής χαρακτηριστικών ασθενούς [18]	17
2.14	Λογότυπο GNU Health [19]	17
2.15	Παράμετροι για την ολοκληρωμένη ιατρική [20]	18
2.16	Πλατφόρμες GNU Health [19]	19
2.17	Οθόνη σύνδεσης [20]	19
2.18	Αρχική Οθόνη ασθενούς στο Tryton [20]	20
2.19	Λίστα Ασθενών [20]	20
2.20	Λογότυπο OpenMRS [21]	21
2.21	Διεπαφή χρήστη της OpenMRS εφαρμογής [21]	22
2.22	Αρχική Οθόνη [21]	22
3.1	Λογότυπο Kubernetes [23]	23
3.2	Kubernetes cluster [23]	24
3.3	Node [23]	25
3.4	kubectl [23]	25
3.5	Deployment [23]	26
3.6	Services [23]	26
3.7	Λογότυπο Helm Charts [24]	27

3.8 Λογότυπο Apache Cassandra [25] . . . . .	28
3.9 Cassandra cluster [25] . . . . .	28
3.10 Αποθήκευση δεδομένων σε ένα Cassandra cluster [25] . . . . .	29
3.11 Partition key [25] . . . . .	30
3.12 Cluster and Partition keys [25] . . . . .	30
3.13 Replication Appache Cassandra [25] . . . . .	31
3.14 Mutlidatacenter in Cassandra [25] . . . . .	31
3.15 Λογότυπο Apache Spark [26] . . . . .	32
3.16 Spark Stand-alone [26] . . . . .	33
3.17 Spark Mesos [26] . . . . .	34
3.18 Spark in Kubernetes [26] . . . . .	34
3.19 Λογότυπο React JS [27] . . . . .	35
3.20 DOM [28] . . . . .	36
3.21 Virtual Dom [29] . . . . .	36
3.22 Λογότυπο FastAPI [30] . . . . .	37
3.23 Λογότυπο Nginx [32] . . . . .	39
3.24 Nginx Load Balancer [32] . . . . .	39
3.25 Nginx in Kubernetes [32] . . . . .	40
 4.1 Διάγραμμα Αρχιτεκτονικής Συστήματος . . . . .	42
4.2 Πίνακας patient_temp_registration . . . . .	46
4.3 Πίνακας patients . . . . .	46
4.4 Πίνακας doctor_temp_registration . . . . .	47
4.5 Πίνακας doctors . . . . .	47
4.6 Πίνακας doctor_patient_relations . . . . .	48
4.7 Πίνακας visits . . . . .	49
4.8 Πίνακας shared_visits . . . . .	49
4.9 Πίνακας shared_visits_by_visibility . . . . .	50
4.10 Πίνακας entities . . . . .	51
4.11 Πίνακας patient_analysis . . . . .	51
4.12 Πίνακας spark_jobs . . . . .	52
4.13 Τα endpoints για τον ασθενή . . . . .	54
4.14 Τα endpoints για τον επαγγελματία υγείας . . . . .	55
4.15 Request Body in Swagger . . . . .	56
4.16 Endpoint /patient/total-visits . . . . .	57
4.17 get_current_patient() code . . . . .	58
4.18 User-Fastapi-Cassandra diagram . . . . .	59
4.19 Endpoint /doctor/relationship/create . . . . .	60
4.20 pydantic model . . . . .	60

4.21	Pydantic in request body of endpoint in Swagger . . . . .	61
4.22	Pydantic model in Swagger . . . . .	61
4.23	Είσοδος στην εφαρμογή από τον ασθενή . . . . .	62
4.24	Διαδικασία εγγραφής ασθενή . . . . .	63
4.25	Αρχική σελίδα ασθενή . . . . .	64
4.26	Ενημέρωση ασθενή για ελλείπει στοιχεία προφίλ . . . . .	64
4.27	Συμπληρωμένο προφίλ ασθενούς . . . . .	65
4.28	Μη διαμοιραζόμενη επίσκεψη ασθενούς . . . . .	65
4.29	Διαμοιραζόμενη επίσκεψη ασθενούς . . . . .	66
4.30	Είσοδος επαγγελματία υγεία στην εφαρμογή . . . . .	67
4.31	Βήματα εγγραφής επαγγελματία υγείας . . . . .	68
4.32	Αρχική σελίδα επαγγελματία υγείας . . . . .	68
4.33	Σελίδα προσθήκης ασθενούς . . . . .	69
4.34	Προσθήκη ιστορικών στοιχείων ασθενή . . . . .	69
4.35	Ολοκλήρωση εγγραφής νέου ασθενούς . . . . .	70
4.36	Αναζήτηση ασθενούς . . . . .	70
4.37	Προφίλ ασθενούς . . . . .	71
4.38	Επιπλέον στοιχεία προφίλ από Παροχο/Οργανισμό . . . . .	71
4.39	Επισκέψεις ασθενή . . . . .	72
4.40	Προβολή επίσκεψης ασθενή . . . . .	72
4.41	Δημιουργία επίσκεψης ασθενή . . . . .	73
4.42	Προβολή σελίδας Ιστορικού ασθενή . . . . .	74
4.43	Προβολή επίσκεψης σε άλλο επαγγελματία υγείας(διαμοιραζόμενη επίσκεψη) . . . . .	75
4.44	Διαγράμματα ανάλυσης δεδομένων ασθενή . . . . .	76
4.45	Δημιουργία Layout Ant-design . . . . .	77
4.46	Παράδειγμα κλήσης endpoint στο Frondent . . . . .	78
4.47	Αρχιτεκτονική Εφαρμογών . . . . .	79
4.48	Εικονικές Μηχανές (VMS) . . . . .	80
4.49	Kubernetes cluster . . . . .	82
4.50	Kubernetes cluster (terminal) . . . . .	82
4.51	Enter Caption . . . . .	83
4.52	K8ssandra deployment . . . . .	84
4.53	Pods of K8ssandra in namespace (terminal) . . . . .	84
4.54	Cassandra in Kubernetes . . . . .	85
4.55	Services in namespace my-k8ssandra (terminal) . . . . .	85
4.56	Cassandra service deployment . . . . .	86
4.57	Dockerfile of fastapi . . . . .	87
4.58	Fastapi of doctor deployment file . . . . .	87
4.59	Deployment of service (doctor fastapi) . . . . .	88

4.60	Fastapi service in Kubernetes	88
4.61	Nginx in Kubernetes	89
4.62	Spark setup	90
4.63	Spark UI	91
4.64	Spark analysis pipeline example	93
4.65	Spark pipeline in Kubernetes cluster	94
4.66	Αποτέλεσμα εκτέλεσης του patientInsights.py (Frontend)	94

# Κεφάλαιο 1

## ΕΙΣΑΓΩΓΗ

### 1.1 Ορισμός των Προβλήματος

Στο διαρκώς μεταβαλλόμενο πεδίο της υγειονομικής περίθαλψης, η ζήτηση για βελτιωμένες υπηρεσίες, αποτελεσματικότερη διαχείριση και άμεση πρόσβαση σε ιατρικές πληροφορίες, αυξάνεται, ιδιαίτερα κατά τη διάρκεια έκτακτων καταστάσεων όπως η πανδημία COVID-19. Η εξέλιξη αυτών των αναγκών στον τομέα της υγείας υπογραμμίζει την προτεραιότητα της ενσωμάτωσης νέων τεχνολογικών λύσεων. Η διαρκής επιδίωξη να προσαρμοστούμε σε αυτές τις σύγχρονες προκλήσεις μέσω της ψηφιακής καινοτομίας μας οδηγεί στην εποχή της Ηλεκτρονικής Υγείας (e-Health).

Με τον όρο Ηλεκτρονική Υγεία (e-Health), σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ) και την Ευρωπαϊκή Επιτροπή, περιγράφεται η αξιοποίηση των σύγχρονων τεχνολογιών της Πληροφορικής και των Τηλεπικοινωνιών σε όλο το φάσμα των παρεχόμενων υπηρεσιών από τους επαγγελματίες υγείας. Στο πλαίσιο της Ηλεκτρονικής Υγείας συμπεριλαμβάνονται προϊόντα, συστήματα και υπηρεσίες που υπερβαίνουν τις απλές διαδικτυακές εφαρμογές, απευθυνόμενες τόσο στο ιατρονοσηλευτικό προσωπικό όσο και στους ασθενείς-χρήστες των υπηρεσιών υγείας. Η Ηλεκτρονική Υγεία περιλαμβάνει πληθώρα εφαρμογών όπως, ηλεκτρονικό φάκελο υγείας, ηλεκτρονική κάρτα υγείας, ηλεκτρονική συνταγογράφηση, υπηρεσίες Τηλεϊατρικής, τηλεσυμβουλευτικής και τηλεπαρακολούθησης [1][2].

Η διπλωματική εργασία ασχολείται με την έννοια του Ηλεκτρονικού Φακέλου Υγείας - ΗΦΥ (Electronic Health Record - EHR), μια πιο προηγμένη μορφή του παραδοσιακού χειρόγραφου ιατρικού φακέλου. Ο ΗΦΥ προσφέρει μια νέα προοπτική στον τρόπο με τον οποίο αποθηκεύονται, διαχειρίζονται και ανακτώνται τα ιατρικά δεδομένα μέσα από σύγχρονα πληροφοριακά συστήματα. Τα ψηφιακά πλέον αρχεία υγείας έχουν την δυνατότητα να βελτιώνουν την ποιότητα, αποτελεσματικότητα και ασφάλεια της παροχής υγειονομικής περίθαλψης. Ωστόσο, παρά τα οφέλη των συστημάτων ΗΦΥ, υπάρχουν και κάποιες προκλήσεις που πρέπει να αντιμετωπιστούν:

- **Η αποκεντρωμένη δομή των συστημάτων**, δηλαδή τα δεδομένα και οι διαδικασίες

είναι διασκορπισμένες σε διαφορετικά συστήματα.

- **Η έλλειψη συνοχής μεταξύ των συστημάτων**, δηλαδή τα συστήματα αυτά δεν είναι ενοποιημένα και δεν μοιράζονται τα ίδια δεδομένα και διαδικασίες.
- **Η αδυναμία ανταλλαγής πληροφοριών** μεταξύ διαφορετικών συστημάτων.

Τα παραπάνω μπορούν να οδηγήσουν σε μια σειρά προβλημάτων όπως:

- δυσκολία στην πρόσβαση και την επεξεργασία δεδομένων
- δυσκολία στην κοινή χρήση δεδομένων μεταξύ διαφορετικών συστημάτων
- δυσκολία στην ασφάλεια καθώς τα δεδομένα μπορεί να είναι πιο ευάλωτα σε επιθέσεις εάν βρίσκονται σε διαφορετικά συστήματα
- δυσκολία στη συντήρηση και αναβάθμιση των συστημάτων
- δυσκολία στη λήψη αποφάσεων από τους επαγγελματίες υγείας εφόσον δεν έχουν πρόσβαση σε όλες τις σχετικές πληροφορίες

Γίνεται επομένως αντιληπτό ότι καθώς η υιοθέτηση συστημάτων ΗΦΥ αυξάνεται, παράλληλα αυξάνεται και η ανάγκη για πιο ισχυρές, επεκτάσιμες και κατανεμημένες υποδομές για τη διαχείριση των αυξανόμενων δεδομένων και την αποτελεσματικότερη παροχή υγειονομικής περίθαλψης.

## 1.2 Αντικείμενο εργασίας

Αντικείμενο της διπλωματικής εργασίας αποτελεί ο σχεδιασμός και η υλοποίηση ενός ολοκληρωμένου, ευέλικτου, ασφαλούς, και αξιόπιστου συστήματος διαχείρισης ΗΦΥ. Το σύστημα αυτό στοχεύει στην αντιμετώπιση των προβλημάτων των συστημάτων ΗΦΥ που αναφέρθηκαν παραπάνω, προσφέροντας μια ολοκληρωμένη λύση που επικεντρώνεται στην παράλληλη και κατανεμημένη διαχείριση των ιατρικών δεδομένων με σκοπό την βελτίωση της ποιότητας της ιατρικής φροντίδας. Λόγω της πολυπλοκότητας και της μεγάλης κλίμακας του συστήματος, απαιτήθηκε η αξιοποίηση μιας ποικιλίας τεχνολογιών, καθεμία από τις οποίες έχει το δικό της ρόλο στην αρχιτεκτονική του συστήματος. Χρησιμοποιήθηκαν τεχνολογίες που σχετίζονται με την αποθήκευση και επεξεργασία δεδομένων (Cassandra, Apache Spark), την ανάπτυξη των εφαρμογών (FastAPI, ReactJS) καθώς και τα κατάλληλα εργαλεία (Kubernetes, Helm charts, Nginx) για την ενσωμάτωση των παραπάνω ως ένα ενιαίο σύστημα.

Το προτεινόμενο σύστημα θα βασίζεται στις ακόλουθες αρχές:

- Κατανεμημένη αρχιτεκτονική για την βελτίωση της διαθεσιμότητας και την ακρίβεια των δεδομένων.

- Παράλληλα υπολογιστικά συστήματα για την βελτίωση της απόδοση του.
- Ενιαία βάση δεδομένων για τη συνοχή του συστήματος.
- Σύγχρονες τεχνολογίες για την ευελιξία, επεκτασιμότητα, ασφάλεια και αξιοπιστία του συστήματος.
- Προσβασιμότητα και χρηστικότητα προς τους χρήστες του συστήματος.

### 1.3 Συνεισφορά της εργασίας

Η συνεισφορά της εργασίας αυτής είναι η ανάπτυξη ενός νέου συστήματος διαχείρισης ΗΦΥ που έχει τη δυνατότητα να βελτιώσει την αποτελεσματικότητα και την ποιότητα της υγειονομικής περίθαλψης.

Το προτεινόμενο σύστημα θα έχει τα ακόλουθα πλεονεκτήματα:

- Θα καταστήσει δυνατή την ολοκληρωμένη και αποτελεσματική διαχείριση ιατρικών δεδομένων,
- Θα διευκολύνει τον γρήγορο και ασφαλή διαμοιρασμό πληροφοριών μεταξύ των μονάδων υγείας,
- Θα επιτρέψει τη λεπτομερή ανάλυση των δεδομένων για κλινικές αξιολογήσεις και θεραπευτικές αποφάσεις/αγωγές,
- Θα δώσει τη δυνατότητα στους ασθενείς για άμεση και ασφαλή πρόσβαση στα προσωπικά τους ιατρικά αρχεία.

Με αυτόν τον τρόπο, το σύστημα θα αξιοποιεί πλήρως τα δεδομένα για το καλό των ασθενών, των επαγγελματιών υγείας και του ευρύτερου συστήματος υγείας.

### 1.4 Δομή και μεθοδολογία εργασίας

Η μεθοδολογία που ακολουθείται στην παρούσα εργασία είναι η εξής:

1. **Βιβλιογραφική επισκόπηση:** γίνεται επισκόπηση της υπάρχουσας βιβλιογραφίας σχετικά με τα συστήματα ΗΦΥ, τα παράλληλα και κατανεμημένα συστήματα και τις τεχνολογίες και τα εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη του συστήματος. Η βιβλιογραφική επισκόπηση θα βοηθήσει στην κατανόηση των τελευταίων εξελίξεων στο πεδίο και θα διασφαλίσει ότι το σύστημα θα είναι συμβατό με τις τρέχουσες τάσεις.
2. **Σχεδιασμός:** γίνεται μια διεξοδική ανάλυση των απαιτήσεων του συστήματος. Εξετάζονται τα ακόλουθα θέματα:

- Οι χρήστες που θα χρησιμοποιούν το σύστημα.
- Οι λειτουργίες που θα πρέπει να υποστηρίζει το σύστημα.
- Τα δεδομένα που θα πρέπει να αποθηκεύει και να επεξεργάζεται το σύστημα.
- Τα περιβάλλοντα στα οποία θα λειτουργεί το σύστημα.

Με βάση την ανάλυση των απαιτήσεων, στη συνέχεια θα σχεδιαστεί η αρχιτεκτονική του προτεινόμενου συστήματος.

**3. Ανάπτυξη:** αναπτύσσονται οι εφαρμογές του προτεινόμενου συστήματος. Οι εφαρμογές θα αναπτυχθούν χρησιμοποιώντας τις τεχνολογίες και τα εργαλεία που αναφέρονται στη βιβλιογραφική επισκόπηση.

Η εργασία δομείται σε 5 κεφάλαια. Αντιλαμβανόμενοι τις προκλήσεις της σύγχρονης εποχής και των ιδιαιτεροτήτων που την διέπουν, στο Κεφάλαιο 1 επιχειρείται αρχικά μια εισαγωγή στο υπάρχον πρόβλημα με τον εξ αρχής ορισμό του, παρουσιάζεται το αντικείμενο της εργασίας και η συνεισφορά της στον τομέα της Υγείας. Στο Κεφάλαιο 2 παρουσιάζονται τα βασικά στοιχεία του ΗΦΥ, αποτυπώνεται η παρούσα κατάσταση στην χώρα μας και γίνεται μια αναφορά από 3 διεθνή συστήματα διαχείρισης ΗΦΥ ανοιχτού κώδικα - OpenEMR, GNU Health, OpenMRS-. Στο Κεφάλαιο 3 αναλύονται όλες οι τεχνολογίες και τα εργαλεία που θα χρησιμοποιηθούν στην υλοποίηση της εφαρμογής του ΗΦΥ, μέσα από την βιβλιογραφική ανασκόπηση. Στο Κεφάλαιο 4 παρουσιάζεται λεπτομερώς η αρχιτεκτονική, η σχεδίαση και η υλοποίηση του προτεινόμενου Ηλεκτρονικού Φακέλου Υγείας, που αποτελεί και το βασικό αντικείμενο της Διπλωματικής εργασίας. Η σύνοψη της εργασίας, τυχόν περιορισμοί που προέκυψαν κατά την υλοποίηση της και οι απόψεις αναφορικά με μελλοντικές επεκτάσεις της, αναγράφονται στο Κεφάλαιο 5. Τέλος, παρουσιάζεται η βιβλιογραφία που χρησιμοποιήθηκε στο πλαίσιο της εργασίας.

## Κεφάλαιο 2

# ΗΛΕΚΤΡΟΝΙΚΟΣ ΦΑΚΕΛΟΣ ΥΓΕΙΑΣ

### 2.1 Η εμφάνιση του ΗΦΥ

Ο τομέας της υγειονομικής περίθαλψης αναπτύσσεται διαρκώς με βάση την πρόοδο της επιστήμης, της τεχνολογίας και των κοινωνικών αναγκών. Μία από τις πιο μεταμορφωτικές αλλαγές στη σύγχρονη υγειονομική περίθαλψη ήταν η μετάβαση προς την ψηφιοποίηση με την εισαγωγή και υιοθέτηση των Ηλεκτρονικών Συστημάτων Υγείας - (Electronic Health System – EHS) και ιδιαίτερα την εισαγωγή της έννοιας του Ηλεκτρονικού Ιατρικού Φακέλου – ΗΙΦ - (Health Medical Record – EMR). Τα ιατρικά στοιχεία ξεκίνησαν να αποθηκεύονται σε πληροφοριακά συστήματα προσφέροντας άμεση και εύκολη ανάκληση τους οποιαδήποτε στιγμή.

Οι τεχνολογικές εξελίξεις καθώς και η απαίτηση για βελτιωμένες υπηρεσίες υγείας, αποδοτικότερη διαχείριση, άμεση και ακριβής πρόσβαση σε ιατρικές πληροφορίες επέβαλλαν την αναβάθμιση του από ένα απλό σύστημα καταγραφής ιατρικών στοιχείων σε ένα πιο ολοκληρωμένο, δομημένο σύστημα που να επιτρέπει την καθολική διαχείριση των ιατρικών στοιχείων μέσα από τον υπολογιστή. Έτσι, ο **Ηλεκτρονικός Ιατρικός Φάκελος** αναβαθμίστηκε σε **Ηλεκτρονικό Φάκελο Υγείας** και άρχισε να χρησιμοποιείται όχι αποσπασματικά μόνο κατά τη νοσηλεία του ασθενή αλλά για την συνεχή παρακολούθηση της υγείας του [3].

### 2.2 Ορισμός και χαρακτηριστικά ΗΦΥ

Ο ΗΦΥ σε απλά λόγια αποτελεί την συστηματική συλλογή του ιστορικού και της κατάστασης υγείας του ασθενή σε ηλεκτρονική μορφή. Τα στοιχεία που καταγράφονται στον ΗΦΥ είναι διαφόρων ειδών και μπορούν να περιλαμβάνουν κλινικά δεδομένα όπως φαρμακευτικές αγωγές, κλινικές εξετάσεις, οικογενειακό ιστορικό, παρατηρήσεις σχετικά με την πρόοδο της αποθεραπείας, οδηγίες πρόληψης, αποτελέσματα εργαστηριακών εξετάσεων, εμβόλια, ζωτικά σημεία και αναφορές απεικονιστικών εξετάσεων [4]. Αποτελεί έναν φάκελο ο οποίος μπορεί να διανεμηθεί τόσο μεταξύ διαφόρων ειδικοτήτων επαγγελματιών υγείας,

όσο και μεταξύ διαφορετικών φορέων υγειονομικής περίθαλψης.

Χαρακτηρίζεται ως μια διαχρονική καταγραφή πληροφοριών υγείας, η οποία επιτυγχάνεται με την διασύνδεση διαφορετικών συστημάτων που συλλέγουν πληροφορίες και στοιχεία υγείας των πολιτών. Αυτό το σύνθετο δίκτυο καταγραφής δεδομένων αποτελείται από άτομα, πληροφορίες, κατευθυντήριες οδηγίες και μεθόδους, καθώς και συστήματα επεξεργασίας, αποθήκευσης και επικοινωνίας [5].

Σύμφωνα με τον οργανισμό HIMSS (Healthcare Information and Management Systems Society) ο ΗΦΥ διαθέτει τα ακόλουθα χαρακτηριστικά [6][7]:

- «*ο ΗΦΥ είναι ένας επεκτάσιμος ηλεκτρονικός φάκελος με πληροφορίες υγείας για έναν ασθενή, οι οποίες παράγονται από έναν ή περισσότερους συμμετέχοντες σε οποιοδήποτε σύστημα παροχής υπηρεσιών φροντίδας υγείας.*
- «*οι πληροφορίες που συμπεριλαμβάνει μεταξύ άλλων είναι τα δημογραφικά στοιχεία του ασθενή, σημειώσεις προόδου, προβλήματα, φάρμακα, ζωτικά σημεία, ιατρικό ιστορικό, εμβόλια, αποτελέσματα εργαστηριακών εξετάσεων και αναφορές απεικονιστικών εξετάσεων.*
- «*ο ΗΦΥ αυτοματοποιεί και βελτιώνει τη ροή της εργασίας των κλινικών ιατρών. Έχει ακόμη τη δυνατότητα να παράγει πλήρη πρακτικά για τον κλινικό ασθενή, καθώς επίσης και να ενισχύει κι άλλες σχετικές με τη φροντίδα του ασθενή δραστηριότητες άμεσα ή έμμεσα μέσω της ανάλογης διεπαφής (συμπεριλαμβάνονται η υποστήριξη αποφάσεων βάσει γεγονότων, η διαχείριση ποιότητας και η υποβολή εκθέσεων εκβάσεων)*

Το περιεχόμενο του ΗΦΥ εμπεριέχει ποικιλία διαφορετικών λειτουργιών, τα οποία διαφοροποιούνται ανάλογα με το σύστημα άλλα και την χώρα στην οποία λειτουργεί. Σύμφωνα με τους Fosse & Dorfmanl [8] το περιεχόμενο του ΗΦΥ μπορεί να διακριθεί σε 5 βασικές κατηγορίες:

- πληροφορίες υγείας
- σύστημα υποστήριξης κλινικών αποφάσεων
- καταχώρηση και διαχείριση απαιτούμενων και διενεργηθεισών εξετάσεων
- απεικονιστικά στοιχεία
- στοιχεία διοικητικού και διαχειριστικού χαρακτήρα

Ο ΗΦΥ ως εργαλείο ενός ψηφιακού συστήματος υγείας που περιέχει πληροφορίες σχετικά με την υγειονομική κατάσταση των ασθενών, θα πρέπει να διαθέτει συγκεκριμένα χαρακτηριστικά όπως [5]:

**1. Δια-συνδεσιμότητα:**

Η δυνατότητα διανομής και ανταλλαγής δεδομένων με άλλα συστήματα, εξασφαλίζοντας κοινά περιβάλλοντα για όλους τους χρήστες όπου τα δεδομένα θα είναι αναγνώσιμα.

**2. Μεταφερσιμότητα (portability):**

Η δυνατότητα μεταφοράς δεδομένων από οργανισμό σε οργανισμό, ανεξαρτήτως του λογισμικού, του εξοπλισμού ή της μητρικής γλώσσας που χρησιμοποιείται.

**3. Επεκτασιμότητα (extendibility):**

Η δυνατότητα για αναβάθμιση της ικανότητας διεκπεραίωσης του συστήματος σε καθεστώς προσθήκης νέων πόρων.

**4. Ασφάλεια -ατομικότητα:**

Η προστασία του περιεχομένου μέσω της εφαρμογής εξουσιοδοτημένης πρόσβασης και καταγραφής των δράσεων βάσει χρονολογικής σειράς.

**5. Διαθεσιμότητα (availability):**

Η διασφάλιση της συνεχούς λειτουργίας του συστήματος όλο το εικοσιτετράωρο και όλες της ημέρες της εβδομάδας.

**6. Εξέλιξη:**

Η δυνατότητα ενημέρωσης στις πιο σύγχρονες εκδόσεις λογισμικού.

**7. Διαχρονική συμβατότητα:**

Η δυνατότητα υποστήριξης του ιατρικού φακέλου για μεγάλα χρονικά διαστήματα.

**8. Χρήση προτύπων:**

Η δυνατότητα χρήσης συγκεκριμένων προτύπων για τον καθορισμό της κοινής δομής της πληροφορίας και των κοινών χαρακτηριστικών για κάθε αυτοματοποιημένο ιατρικό φάκελο.

## 2.3 Ηλεκτρονικός Φάκελος Υγείας/Ηλεκτρονικός Ιατρικός Φάκελος

Από το 1960 που πρωτοεμφανίστηκε η ιδέα του φακέλου υγείας έως και σήμερα έχει χρησιμοποιηθεί πλήθος ακρωνύμιων προκειμένου να τους κατηγοριοποιήσει και να επισημάνει τις διαφορές τους. Οι ορισμοί αυτοί συνήθως είναι αμφιλεγόμενοι και καλύπτουν ένα μεγάλο εύρος, με αποτέλεσμα να δημιουργούν πολλές φορές λανθασμένη χρήση αυτών. Σύμφωνα με το Medical Records Institute διακρίνονται πέντε επίπεδα ηλεκτρονικών φακέλων φροντίδας υγείας (EHCNs – Electronic Health Care Records) ξεκινώντας από το χαμηλότερο και φτάνοντας στο υψηλότερο επίπεδο [7]:

- Automated Medical Record (AMR)
- Computerized Medical Record (CMR)
- Electronic Medical Record (EMR)
- Electronic Patient Records (EPR)
- Electronic Health Records (EHR)

Σε αυτό το σημείο κρίνεται σκόπιμο να γίνει αποσαφήνιση μεταξύ των όρων «Ηλεκτρονικός Φάκελος Υγείας (ΗΦΥ) ή Electronic Health Record (EHR)» και «Ηλεκτρονικός Ιατρικός Φάκελος (ΗΙΦ) ή Electronic Medical Record (EMR)», που ενώ φαινομενικά μπορεί να θεωρηθούν ισοδύναμοι, εντούτοις αποτελούν ξεχωριστές έννοιες.

Ειδοποιός διαφορά μεταξύ αυτών των όρων έγκειται στην διαφορετική ερμηνεία των λέξεων ”υγεία” (health) και ”ιατρική” (medical). Ο πρώτος όρος που χρησιμοποιήθηκε κλινικά ήταν αυτός του EMR που περιλαμβάνει την λέξη ”ιατρική” (medical) και αφορούσε τη διάγνωση, τη θεραπεία και την πρόληψη ασθενειών και καταστάσεων. Αντιθέτως, σύμφωνα με τον ΠΟΥ ο ορισμός της ”υγείας” που δίνεται αναφέρει ότι «υγεία είναι όχι απλώς η έλλειψη ασθένειας, αλλά η πλήρης σωματική, κοινωνική και ψυχική ευεξία του ατόμου». Γίνεται επομένως αντιληπτό πως η λέξη ”υγεία” είναι έννοια δυναμική και πολυδιάστατη καθώς καλύπτει ένα ευρύτερο φάσμα θεμάτων σε σχέση με τη λέξη ”ιατρική” (medical) [9].

Ο ΗΙΦ περιλαμβάνει ηλεκτρονικά ιατρικά αρχεία (ιατρικό και θεραπευτικό ιστορικό) ασθενών τα οποία διατίθενται στο αρχείο του κλινικού ιατρού και δεν μπορούν να διανεμηθούν εύκολα. Η εφαρμογή στην οποία είναι αποθηκευμένες αυτές οι πληροφορίες δεν είναι προσβάσιμη από άλλους επαγγελματίες υγείας, οργανισμούς ή τους ασθενής, με αποτέλεσμα τα αρχεία να πρέπει να εκτυπωθούν για να μπορέσουν να μεταφερθούν [9].

Αντίθετα ο ΗΦΥ περιλαμβάνει πληροφορίες με την συνολική εικόνα της υγείας του ασθενή, πέρα από τα κλινικά δεδομένα που συλλέγονται στο γραφείο του παρόχου. Έχει σχεδιαστεί για το διαμοιρασμό των πληροφοριών με άλλους παρόχους υγειονομικής περίθαλψης, όπως εργαστήρια και ειδικούς, έτσι ώστε να περιέχουν πληροφορίες από όλους τους κλινικούς γιατρούς που εμπλέκονται στη φροντίδα του ασθενούς. Συμπερασματικά έχουν σχεδιαστεί για να παρέχουν πρόσβαση σε όλα τα άτομα που εμπλέκονται στη φροντίδα των ασθενών - συμπεριλαμβανομένων των ίδιων των ασθενών [9].

Σε μια πιο περιγραφική μορφή δίνονται οι ορισμοί σύμφωνα με τον HIMSS [10]:  
**«Ηλεκτρονικός Ιατρικός Φάκελος (EMR):** Ενα ηλεκτρονικό αρχείο πληροφοριών που σχετίζονται με την υγεία για ένα άτομο, το οποίο μπορεί να δημιουργηθεί, να συλλεχθεί, να διαχειριστεί και να συμβουλευτεί εξουσιοδοτημένους κλινικούς ιατρούς και προσωπικό σε έναν οργανισμό υγειονομικής περίθαλψης.»

**«Ηλεκτρονικός φάκελος υγείας (EHR):** Ενα ηλεκτρονικό αρχείο πληροφοριών που σχετίζεται με την υγεία για ένα άτομο, το οποίο συμμορφώνεται με εθνικά αναγνωρισμένα πρότυπα

*διαλειτουργικότητας και το οποίο μπορεί να δημιουργηθεί, να διαχειρίζεται και να συμβουλεύεται εξουσιοδοτημένους κλινικούς ιατρούς και προσωπικό σε περισσότερους από έναν οργανισμός υγειονομικής περίθαλψης.»*

Συμπερασματικά, γίνεται κατανοητό ότι ο ΗΦΥ αποτελεί ένα υπερσύνολο του ΗΙΦ.

## 2.4 Πλεονεκτήματα Ηλεκτρονικού Φακέλου Υγείας

Οπως αναφέρθηκε και παραπάνω ο ΗΦΥ είναι ένας ολοκληρωμένος ιατρικός φάκελος δίνοντας την δυνατότητα στον ασθενή και στους επαγγελματίες υγείας να έχουν πρόσβαση σε όλα τα δεδομένα που αφορούν την υγεία και γενικότερα την υγειονομική περίθαλψη του ασθενή. Ο ΗΦΥ έχει φέρει επανάσταση στη σύγχρονη υγειονομική περίθαλψη με πολλούς τρόπους, αμβλύνοντας τα μειονεκτήματα των παραδοσιακών ιατρικών αρχείων και βελτιώνοντας την υγειονομική περίθαλψη. Τα οφέλη από την χρήση των ΗΦΥ είναι πολλαπλά. Ακολούθως, παρουσιάζονται ενδεικτικά κάποια από τα βασικά πλεονεκτήματα, όπως αυτά προέκυψαν μέσα από την βιβλιογραφική αναζήτηση [11][12]:

- Καλύτερη Ποιότητα Φροντίδας
- Άμεση εμφάνιση και κοινή πρόσβαση στους μακροχρόνιους φακέλους υγείας ενός ασθενούς από οπουδήποτε και οποτεδήποτε αυτό είναι απαραίτητο
- Ασφαλής ανταλλαγή ηλεκτρονικών πληροφοριών με ασθενείς και άλλους επαγγελματίες υγείας
- Δυνατότητα γρήγορης πρόσβασης στα αρχεία ασθενών για πιο συντονισμένη και αποτελεσματική φροντίδα
- Προβολή προηγούμενων αποφάσεων, με σκοπό την αποτελεσματικότερη παροχή υγειονομικής φροντίδας και την καλύτερη ποιότητα νοσηλείας και εξυπηρέτησης
- Βελτίωση της αλληλεπίδρασης και της επικοινωνίας ασθενών και επαγγελματιών υγείας
- Παροχή ακριβών, ενημερωμένων και πλήρεις πληροφοριών για τους ασθενείς
- Ενεργοποίηση ασφαλέστερης, πιο αξιόπιστης συνταγογράφησης
- Βελτίωση του απορρήτου και της ασφάλειας των δεδομένων ασθενών
- Μείωση των περιττών ιατρικών εξετάσεων με άμεσο αποτέλεσμα τη μείωση της επιβάρυνσης της υγείας των ασθενών από επίπονες και βλαπτικές εξετάσεις, καθώς και τη μείωση του τελικού κόστους.
- Μείωση της γραφειοκρατίας και επομένως μείωση του κόστους.

## 2.5 Ηλεκτρονικός Φάκελος Υγείας στην Ελλάδα

Ο ΗΦΥ έρχεται αρκετά αργοπορημένα στο ελληνικό σύστημα υγείας, μόλις πριν λίγα χρόνια με τον Ν.4486/2017 (ΦΕΚ 115/Α/7-8-17) «Μεταρρύθμιση της Πρωτοβάθμιας Φροντίδας Υγείας, επείγουσες ρυθμίσεις αρμοδιότητας Υπουργείου Υγείας και άλλες διατάξεις» και συγκεκριμένα από το άρθρο 21, ο οποίος αργότερα αντικαταστάθηκε με τον Ν. 4600/2019, «Εκσυγχρονισμός και Αναμόρφωση Θεσμικού Πλαισίου Ιδιωτικών Κλινικών, Σύσταση Εθνικού Οργανισμού Δημόσιας Υγείας, Σύσταση Εθνικού Ινστιτούτου Νεοπλασιών και λοιπές διατάξεις», (ΦΕΚ 43/Α/09-03-2019) [13][14].

Σύμφωνα με την νομοθεσία κάθε πολίτης με ΑΜΚΑ θα πρέπει να διαθέτει Ατομικό Ηλεκτρονικό Φάκελο Υγείας (ΑΗΦΥ) όπως τον ονομάζει, ο οποίος θα περιλαμβάνει όλα τα έγγραφα που περιέχουν δεδομένα, εκτιμήσεις και πληροφορίες κάθε είδους σχετικά με την κατάσταση και την κλινική εξέλιξη του ασθενούς καθ' όλη τη διαδικασία περίθαλψης. Πρόσβαση σε αυτόν θα έχουν αρχικά ο κάθε πολίτης με ΑΜΚΑ, ο οικογενειακός ιατρός ή άλλος επαγγελματίας υγείας, κατά τη νοσηλεία ή επίσκεψη σε δημόσια ή ιδιωτική μονάδα παροχής υπηρεσιών υγείας, και ύστερα από συναίνεση του ατόμου. Επίσης, σύμφωνα με τα όσα ορίζει η νομοθεσία, με σχετική απόφαση του Υπουργείου Υγείας καθιερώνεται ενιαίο πρότυπο αναφορικά με το περιεχόμενο, τις διαδικασίες κατάρτισης και ταυτοποίησης του ατόμου, αλλά και της πρόσβασης στις ιατρικές πληροφορίες του φακέλου, το περιεχόμενο του οποίου είναι ενιαίο σε εθνικό επίπεδο και υποχρεωτικό [15].

Στην Ελλάδα ο ΑΗΦΥ διεκπεραιώνεται μέσω της ηλεκτρονικής πλατφόρμας «**ΗΔΙΚΑ**», η οποία δημιουργήθηκε από την εταιρεία με την επωνυμία «ΗΛΕΚΤΡΟΝΙΚΗ ΔΙΑΚΥΒΕΡΝΗΣΗ ΚΟΙΝΩΝΙΚΗΣ ΑΣΦΑΛΙΣΗΣ ΑΕ». Μέσα από αυτή την εφαρμογή δίνεται η δυνατότητα στην ιατρική κοινότητα και στους πολίτες, να έχουν πλήρη πρόσβαση στο σύνολο των ιατρικών δεδομένων. Ο ΑΗΦΥ αντλεί στοιχεία από διάφορα συστήματα υγείας (ΕΟΠΥΥ, ΠΦΥ, Ηλεκτρονική Συνταγογράφηση, Νοσοκομεία, Διαγνωστικά Κέντρα, Ιδιώτες ιατρούς, ιδιωτικές ΜΥ, Κεντρικό αποθετήριο ιατρικών εξετάσεων, κεντρικό αποθετήριο απεικονιστικών εξετάσεων, μητρώα ασθενών, κ.λ.π.).

Προκειμένου κάθε πολίτης να έχει πρόσβαση στον ΑΗΦΥ του θα πρέπει να κάνει εγγραφή στην ηλεκτρονική εφαρμογή μέσω της εφαρμογής της Πρωτοβάθμιας Φροντίδας Υγείας [16].

Αφού επιλεγεί ο τρόπος εισόδου στην εφαρμογή και γίνει η αντίστοιχη αυθεντικοποίηση των στοιχείων, εμφανίζεται η οθόνη εισαγωγής του ΑΜΚΑ του χρήστη.

Αφού εισέλθει ο χρήστης στην εφαρμογή στην αρχική οθόνη εμφανίζονται οι εξής επιλογές:

- Ατομικά στοιχεία
- Άυλη Συνταγογράφηση
- Νέα Ραντεβού

The screenshot shows the govgr login page. At the top, there's a logo for govgr BETA and the text "Σύστημα Πρωτοβάθμιας Φροντίδας Υγείας". On the right, there's a logo for ΗΔΙΚΑ (Ηλεκτρονική Διακυβέρνηση Κοινωνικής Ασφαλίσης). Below the header, there are two rows of buttons for login methods:

- Row 1:**
  - Είσοδος με κωδικούς TaxisNet και OTP (Προσωπικός Ιατρός και όλες τις άλλες υπηρεσίες)
  - Είσοδος με κωδικούς TaxisNet (Αυλη Συνταγογράφηση, Ηλεκτρονικά Ραντεβού)
  - Είσοδος με κωδικούς ΑΗΦΥ
  - Είσοδος με eIDAS
- Row 2:** Είσοδος στην υπηρεσία

Below these rows, there's a section titled "Γενικές Πληροφορίες" with tabs for "Αυλη Συνταγογράφηση", "ΑΗΦΥ - Ηλεκτρονικό Βιβλιάριο Υγείας Παιδιού", and "Αναβάθμιση Ασφάλειας Συστήματος". A note below the tabs says: "Για την είσοδο σας στις υπηρεσίες του Συστήματος Πρωτοβάθμιας φροντίδας Υγείας ακολουθήστε τις παρακάτω επιλογές." A bulleted list follows:

- Για την είσοδο στα Ηλεκτρονικά Ραντεβού και την Ενεργοποίηση της Αυλης Συνταγογράφησης
  - Επιλέξτε "Είσοδος με Κωδικούς TaxisNet"
  - Επιλέξτε "Είσοδος με Κωδικούς TaxisNet και OTP" (απαιτείται η δήλωση του κινητού στο ΕΜΕΠ)
- Για την είσοδο στον ΑΗΦΥ, στα Ηλεκτρονικό Βιβλιάριο Υγείας Παιδιού και στον Προσωπικό Ιατρό
  - Επιλέξτε "Είσοδος με Κωδικούς TaxisNet και OTP" (απαιτείται η δήλωση του κινητού στο ΕΜΕΠ)
  - Επιλέξτε "Είσοδος με Κωδικούς ΑΗΦΥ" (δημιουργούνται κατά την εγγραφή σε Προσωπικό Ιατρό)

**Εικόνα 2.1:** Κεντρική σελίδα (login) της εφαρμογής της Πρωτοβάθμιας Φροντίδας Υγείας [16]



**Εικόνα 2.2:** Εισαγωγή ΑΜΚΑ [16]

The screenshot shows the govgr BETA website interface. At the top, there are navigation links: Απομική Στοιχεία, Άυλη Συνταγογράφηση, Νέο Ραντεβού, Ηλεκτρονικές Υπηρεσίες, Ηλεκτρονικός Φάκελος Υγείας, and Με μια ματιά. On the right, there is a logo for ΗΔΙΚΑ (ΕΠΟΧΡΗΜΑΤΙΚΗ ΕΠΙΧΕΙΡΗΣΗ ΚΛΗΡΟΝΟΜΙΩΝ Α.Ε.) and a button labeled 'Αποσύνδεση'.

**Me mia matia**

**Ανακοινώσεις**

Ημερομηνία	Ανακοίνωση
07/07/2022	Σας ενημερώνουμε ότι η διαδικασία εγγραφής σε προσωπικό ιατρό δεν λειτουργεί προσωρινά, αναμένεται να ξεκινήσει τις επόμενες ημέρες. Θα ακολουθήσει νεώτερη ανακοίνωση σχετικά.
04/07/2022	Κατόπιν σχετικής οδηγίας του Υπουργείου Υγείας σας ενημερώνουμε ότι από 04/07/2022 οι Συμβεβλημένοι Ιατροί του ΕΟΠΥΥ και οι Παιδίατροι που έχουν οριστεί ως Οικογενειακοί Ιατροί παύουν προσωρινά να είναι άριστοι προς το κοινό στην εφαρμογή της ΠΓΥ γιατί να μην θέλουν άλλες εγγραφές.

Σας ενημερώνουμε ότι η διαδικασία εγγραφής σε προσωπικό ιατρό δεν λειτουργεί προσωρινά, αναμένεται να ξεκινήσει τις επόμενες ημέρες. Θα ακολουθήσει νεώτερη ανακοίνωση σχετικά.

Κατόπιν σχετικής οδηγίας του Υπουργείου Υγείας σας ενημερώνουμε ότι από 04/07/2022 οι Συμβεβλημένοι Ιατροί του ΕΟΠΥΥ και οι Παιδίατροι που έχουν οριστεί ως Οικογενειακοί Ιατροί παύουν προσωρινά να είναι άριστοι προς το κοινό στην εφαρμογή της ΠΓΥ ώστε να μην θέλουν άλλες εγγραφές.

Σας ενημερώνουμε ότι η διαδικασία εγγραφής σε προσωπικό ιατρό δεν λειτουργεί προσωρινά, αναμένεται να ξεκινήσει τις επόμενες ημέρες. Θα ακολουθήσει νεώτερη ανακοίνωση σχετικά.

**Πληροφορίες Προσωπικού Ιατρού**

Μονάδα Υγείας ιατρού: ΕΠΑΝΑΣΤΑΣΗ ΛΑΖΑΡΟΥ

**Άυλη Συνταγογράφηση**

Δεν έχετε επιλέξει Άυλη Συνταγογράφηση. Εάν το επιθυμείτε, πατήστε "Ενεργοποίηση".

**Ενεργοποίηση**

**Πληροφορίες Μονάδας Υγείας**

**Τα ραντεβού μου**

Ημερομηνία	Μονάδα Υγείας	Ιατρός	Ιατρός	Τύπος ραντεβού	Κατάσταση

**Εικόνα 2.3:** Αρχική οθόνη εφαρμογής [16]

- Ηλεκτρονικός Φάκελος Υγείας
- Με μια ματιά

Επιλέγοντας τον Ηλεκτρονικό Φάκελο Υγείας προσφέρονται στον χρήστη τρεις δυνατότητες:

1. Προβολή Φακέλου
2. Ιστορικό Πρόσβασης
3. Συνοπτικό Ιστορικό Υγείας

The screenshot shows the govgr BETA website interface. At the top, there are navigation links: Απομική Στοιχεία, Άυλη Συνταγογράφηση, Νέο Ραντεβού, Εφαρμογή Προσωπικού Ιατρού, Ηλεκτρονικός Φάκελος Υγείας, and Με μια ματιά. On the right, there is a logo for ΗΔΙΚΑ (ΕΠΟΧΡΗΜΑΤΙΚΗ ΕΠΙΧΕΙΡΗΣΗ ΚΛΗΡΟΝΟΜΙΩΝ Α.Ε.) and a button labeled 'Αποσύνδεση'.

**Ηλεκτρονικός Φάκελος Υγείας**

- Προβολή Φακέλου (Α.Η.Φ.Υ.)
- Ιστορικό Πρόσβασης
- Συνοπτικό Ιστορικό Υγείας

**Εικόνα 2.4:** Επιλογή ΗΦΥ [16]

1. Προβολή Φακέλου

Επιλέγοντας την προβολή Φακέλου ο χρήστης έχει την δυνατότητα να δει όλα τα δεδομένα που έχουν καταχωρηθεί. Η οθόνη αυτή αποτελείται από τις παρακάτω ενότητες, οι οποίες εμφανίζονται στο μενού πλοϊγησης κάτω από τα στοιχεία του ασθενούς:

- Επισκέψεις
- Ραντεβού

- Παιδιατρικό Ιστορικό
- Ατομικό Ιστορικό
- Οικογενειακό Ιστορικό
- Διαγνώσεις
- Φάρμακα
- Εξετάσεις
- Εμβόλια
- Νοσηλείες
- Κοινωνικές Συνήθειες
- Γυναικολογικό Ιστορικό (μόνο για γυναίκες)
- Φάκελοι αναπτηρίας
- Διαγράμματα
- Έγγραφα

**Εικόνα 2.5:** ΗΦΥ Ασθενούς-Επισκέψεις [16]

## 2. Ιστορικό Πρόσβασης

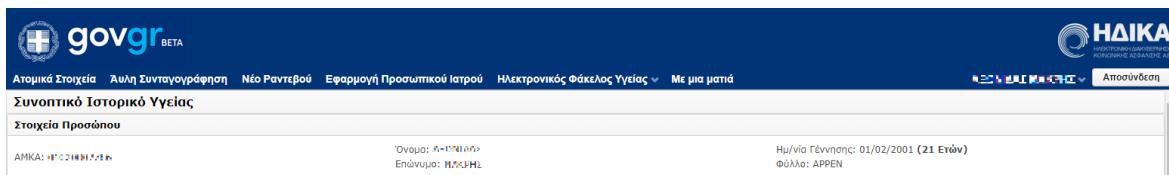
Στην οθόνη του Ιστορικού, ο χρήστης έχει την δυνατότητα να δει ποιος ιατρός και πότε είχε πρόσβαση στον ΗΦΥ του. Στον Πίνακα που εμφανίζεται, δίνονται οι πληροφορίες σχετικά με το ποια ενέργεια πραγματοποίησε ο ιατρός ((Προβολή ή Τροποποίηση Φακέλου), σε ποια ενότητα του φακέλου είχε πρόσβαση και τέλος ποιο ήταν το είδος πρόσβασης στον φάκελο.

**Εικόνα 2.6:** Ιστορικό Πρόσβασης [16]

### 3. Συνοπτικό Ιστορικό Υγείας

Σε αυτήν την οθόνη ο πολίτης μπορεί να δει συνοπτικά τις εξής πληροφορίες σχετικά με το ιστορικό υγείας του:

- Στοιχεία Προσώπου
- Διαγνώσεις των τελευταίων 6 μηνών
- Φαρμακευτική αγωγή των τελευταίων 6 μηνών
- Ενότητες



**Εικόνα 2.7: Στοιχεία Προσώπου [16]**

Διαγνώσεις	
Διαγνώσεις των τελευταίων 6 μηνών	
Κωδικός Νόσου	Περιγραφή
R00	Διατοπαιδείς του κεφαλιού πανσό
E11	Μη νασιλιασθερπάρεσσες σακευράσης διεβήμης
M95	Μάλια, επίθεση, πυρεξιακός, του μαστοκεντρικού φαρμακού, κατά τη συνθήκη της λοιμώ
R80	Γρίπη

**Εικόνα 2.8: Διαγνώσεις [16]**

Φαρμακευτική Αγωγή						
Φαρμακευτική αγωγή των τελευταίων 6 μηνών						
Εμπορική Ονομασία	Δραστική Ουσία	Μονάδα Δόσης	Μορφή Δόσης	Διάρκεια (Ημέρες)	Ημερομηνία Σύνταγμα	Προέλευση
ΟΝΤΡΟ ΚΟΙΝΩΝΙΚΗ ΕΠΙΧΕΙΡΗΣΗ ΤΟΥ ΕΛΛΗΝΙΚΟΥ ΛΑΪΚΟΥ	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	1	ΔΙΣΚΙΑ ΕΛΕΓΧ ΑΠΟΔ.	28	17 Jun, 2022	Συνταγογράφηση
ΑΝΘΡΩΠΙΝΗ ΤΟΝΙΖΟΥΣΑ ΕΠΙΧΕΙΡΗΣΗ ΤΟΥ ΕΛΛΗΝΙΚΟΥ ΛΑΪΚΟΥ	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	2	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	30	08 Jun, 2022	Συνταγογράφηση
ΑΝΘΡΩΠΙΝΗ ΤΟΝΙΖΟΥΣΑ ΕΠΙΧΕΙΡΗΣΗ ΤΟΥ ΕΛΛΗΝΙΚΟΥ ΛΑΪΚΟΥ	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	2	ΚΑΨΟΥΛΑ	15	08 Jun, 2022	Συνταγογράφηση
ΟΝΤΡΟ ΚΟΙΝΩΝΙΚΗ ΕΠΙΧΕΙΡΗΣΗ ΤΟΥ ΕΛΛΗΝΙΚΟΥ ΛΑΪΚΟΥ	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	2	ΔΙΣΚΙΑ ΕΠΙΚΑΛ	30	08 Jun, 2022	Συνταγογράφηση

**Εικόνα 2.9: Φαρμακευτική αγωγή [16]**

Στο πλαίσιο «Ενότητες» εμφανίζονται τα εξής:

- Ατομικό Ιστορικό
- Φάρμακα
- Ειδοποιήσεις
- Κοινωνικό Ιστορικό
- Φυσικά Ευρήματα

- Διαγνωστικές Εξετάσεις

Ενότητες					
Ιατρικό Ιστορικό		Φάρμακα		Ειδησησήσεις	
Διαγνώσεις		Ειδικότητα		Κοινωνικό Ιστορικό	
Διανύσσετες των τελευταίων 12 μηνών				Φυσικά Συρήματα	
Κωδικός Νόσου	Περιγραφή	Ημερομηνία Εμφάνισης	Προέλευση	Κωδικοποίηση	
E01	Απεργούσα ταυτόχρονη απεργία από την ανάπτυξη	04 Jul, 2022	Διάνυσση Επεισοδίου Φραντζίδας	ICD10	
E13	Περιορισμένη διάρροια στην περιοχή της ανάπτυξης	17 Jun, 2022	Συνταγογράφηση	ICD10	
M65	Περιορισμένη διάρροια στην περιοχή της ανάπτυξης	08 Jun, 2022	Συνταγογράφηση	ICD10	
K80	Γρίπη	31 May, 2022	Διάνυσση Επεισοδίου Φραντζίδας	ICPC2	
Z00-Z	Επεισόδιο από την ανάπτυξη	11 Jan, 2022	Συνταγογράφηση	ICD10	
ETB, 5	Χρόνια παραποταμία από την ανάπτυξη	08 Sep, 2021	Συνταγογράφηση	ICD10	
EDL, 5	Οδοντική παραποταμία από την ανάπτυξη	08 Sep, 2021	Συνταγογράφηση	ICD10	
ETD, 5	Υγειακή ιδιομορφισμένη παραποταμία από την ανάπτυξη	18 Aug, 2021	Συνταγογράφηση	ICD10	

Εικόνα 2.10: Ενότητες [16]

## 2.6 Παραδείγματα συστημάτων ΗΦΥ

Στο κεφάλαιο αυτό θα αναλυθούν τρία σημαντικά συστήματα διαχείρισης ΗΦΥ - OpenEMR, OpenMRS, και GNUHealth- που εξυπηρετούν τον σκοπό της συγκεκριμένης διπλωματικής εργασίας. Αυτά τα συστήματα αντιπροσωπεύουν ανοιχτού κώδικα λύσεις που έχουν σχεδιαστεί για να διευκολύνουν την ηλεκτρονική διαχείριση των ιατρικών αρχείων, ενώ παράλληλα προσφέρουν μια σειρά από λειτουργίες που βελτιώνουν την ποιότητα και την αποδοτικότητα της υγειονομικής περίθαλψης.

### 2.6.1 OpenEMR



Εικόνα 2.11: Λογότυπο OpenEMR [17]

Το OpenEMR είναι ένα ανοιχτού κώδικα σύστημα ηλεκτρονικών φακέλων υγείας και διαχείρισης ιατρικής πρακτικής. Αποτελεί ένα από τα διασημότερα συστήματα ανοικτού κώδικα, έχει μεταφραστεί σε 36 γλώσσες και χρησιμοποιείται από εγκαταστάσεις σε περισσότερες από 100 χώρες σε όλο τον κόσμο.

Είναι μια ολοκληρωμένη εφαρμογή που επιτρέπει στους χρήστες -ιατρικό προσωπικό, προσωπικό γραφείου, νοσηλευτικό προσωπικό- να εισάγουν ιατρικά δεδομένα σχετικά με τον κάθε ασθενή. Η εφαρμογή έχει την δυνατότητα να συνδυάζει την ιατρική κλινική διαχείριση και τα ηλεκτρονικά ιατρικά αρχεία και περιλαμβάνει μια μεγάλη γκάμα δυνατοτήτων όπως: δημογραφικά στοιχεία ασθενών, ραντεβού και προγραμματισμός, ηλεκτρονικοί ιατρικοί φάκελοι, διαχείριση κύκλου τιμολόγησης και εσόδων, αναφορές, φαρμακευτικές αγωγές, κλινικούς και παρακλινικούς ελέγχους, πορεία νόσου.

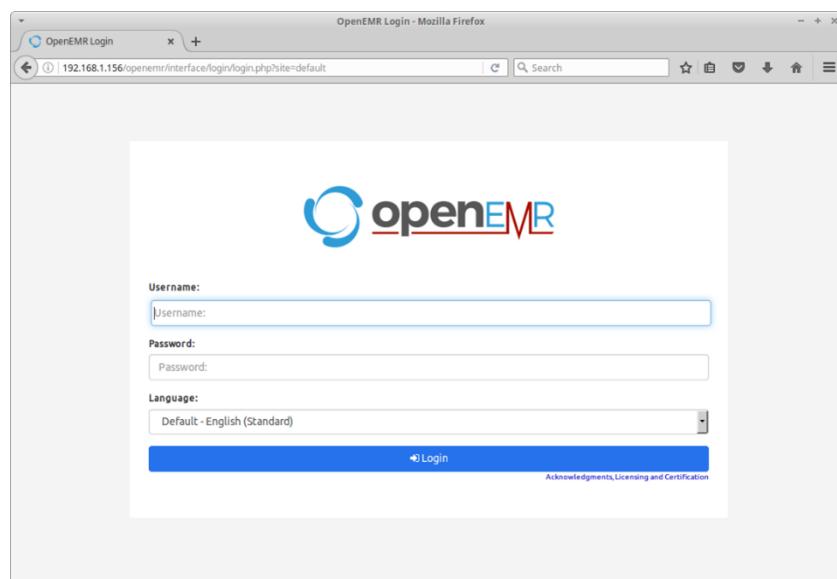
Σε σύγκριση με άλλες ανάλογες εμπορικές εφαρμογές που κοστίζουν εκατοντάδες ευρώ, το OpenEMR όπως αναφέρθηκε διανέμεται δωρεάν μέσω του <https://www.open-emr.org/> και μπορεί να τρέξει στα περισσότερα λειτουργικά συστήματα.

Το OpenEMR είναι πιστοποιημένο από το ONC 2015 Cures Update, πράγμα που σημαίνει ότι πληροί τις απαιτήσεις που έχει θέσει το Γραφείο του Εθνικού Συντονιστή για την Τεχνολογία Πληροφοριών Υγείας (ONC).

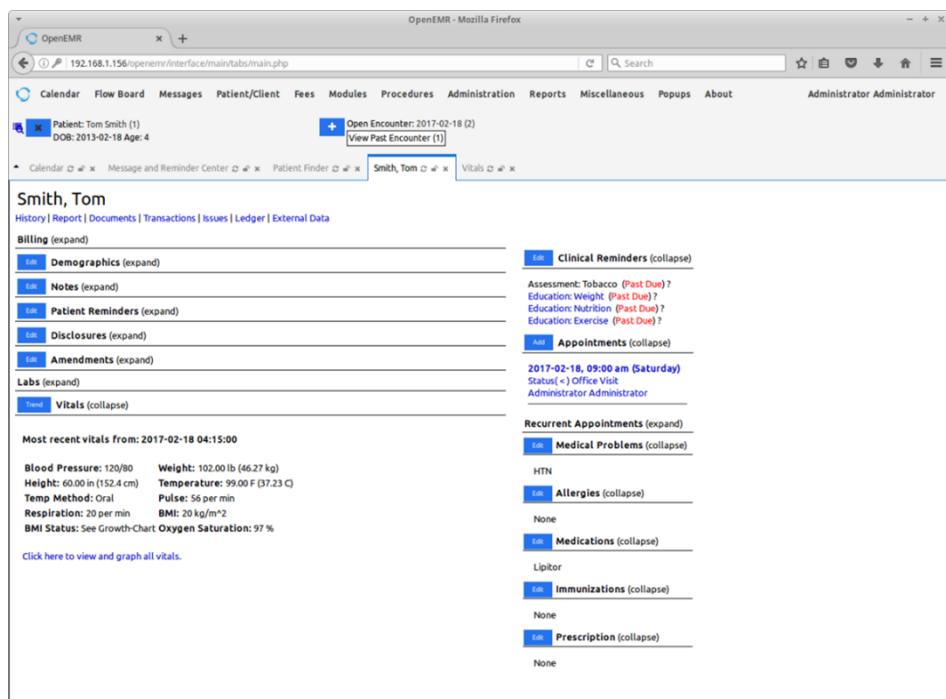
Η αρχιτεκτονική της εφαρμογής είναι τύπου LAMP (Linux, Apache, MySQL, PHP). Συγκεκριμένα η εφαρμογή τρέχει σε έναν διακομιστή Linux με τον Apache ως web server, την MySQL ως βάση δεδομένων, και την PHP ως γλώσσα προγραμματισμού για την ανάπτυξη του κώδικα της.

### **Βασικά Χαρακτηριστικά:**

- **Προσαρμόσιμο:** Όντας ανοιχτού κώδικα, οι κλινικές και τα νοσοκομεία μπορούν να προσαρμόσουν το σύστημα με βάση συγκεκριμένες ανάγκες.
- **Πιστοποιημένο:** Είναι πιστοποιημένο ONC, διασφαλίζοντας ότι πληροί τα καθορισμένα πρότυπα για συστήματα EHR.
- **Εκτεταμένα χαρακτηριστικά:** Περιλαμβάνει προγραμματισμό ασθενών, τιμολόγηση, κανόνες κλινικής απόφασης και υπηρεσίες ηλεκτρονικής χρέωσης.
- **Ενεργή Κοινότητα:** Διαθέτει μια ενεργή κοινότητα προγραμματιστών και χρηστών που παρέχουν συνεχείς ενημερώσεις και βελτιώσεις.



**Εικόνα 2.12:** Αρχική Οθόνη και Είσοδος στο σύστημα [18]



**Εικόνα 2.13:** Καρτέλα καταγραφής χαρακτηριστικών ασθενούς [18]

## 2.6.2 GNU Health

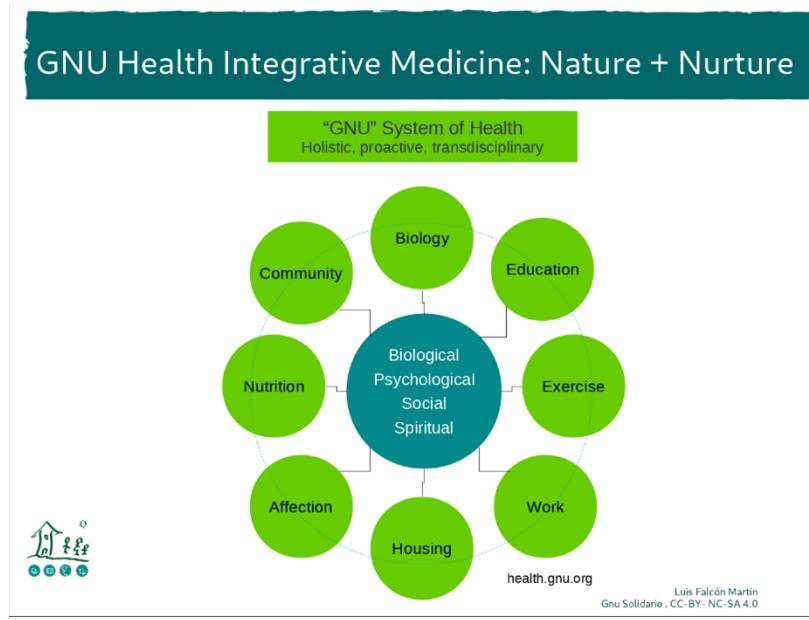


**Εικόνα 2.14:** Λογότυπο GNU Health [19]

Το GNU Health είναι ένα δωρεάν, ανοιχτού κώδικα Σύστημα Πληροφοριών Υγείας και Νοσοκομείου (Health Information System, HIS) που παρέχει μια ολιστική προσέγγιση στη διαχείριση της υγείας, το οποίο το καθιστά πιο ολοκληρωμένο από ένα απλό EMR ή EHR. Βασικός στόχος της εφαρμογής είναι υλοποίηση της έννοιας «Ολοκληρωμένη Ιατρική» (Integrative Medicine). Σχεδιάστηκε για να είναι εξαιρετικά ευέλικτο και μπορεί να προσαρμοστεί για να καλύψει τις ανάγκες των υγειονομικών εγκαταστάσεων όλων των μεγεθών.

Το GNU Health χαρακτηρίζεται ως ένα ελεύθερο έργο που καθοδηγείται από την κοινότητα GNU Solidario, έναν μη κερδοσκοπικό ανθρωπιστικό οργανισμό που επικεντρώνεται στην κοινωνική ιατρική (social medicine) και την πληροφορική της υγείας (health informatics). Στόχος του είναι η ισότιμη πρόσβαση, η συλλογική δουλειά και η αλληλεγγύη.

To GNU Health ξεκίνησε με σκοπό τη βελτίωση των συνθηκών και την συμπλήρωση



**Εικόνα 2.15:** Παράμετροι για την ολοκληρωμένη ιατρική [20]

της πρωτοβάθμιας υγειονομικής περίθαλψης (Primary Health Care, PHC) στις αγροτικές κοινότητες. Το στοιχείο που το έκανε να ξεχωρίσει μεταξύ άλλων είναι η ικανότητα του για διαχείριση και επεξεργασία μεγάλων ποσοτήτων δεδομένων.

Το GNU Health έχει σχεδιαστεί με τέτοιο τρόπο ώστε να μπορεί να εγκατασταθεί σε διάφορα λειτουργικά συστήματα (GNU / Linux, FreeBSD) και σε διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων (PostgreSQL). Είναι γραμμένο στην γλώσσα Python και στο πλαίσιο (framework) Tryton. Σχετικά με το θέμα της ασφάλειας, το GNU Health την έχει εξασφαλίσει μέσω της τεχνολογίας GNUPG. Το GNUPG είναι ένα εργαλείο του GNU Health που επιτρέπει την κρυπτογράφηση και την ψηφιακή υπογραφή των δεδομένων και των επικοινωνιών των χρηστών του συστήματος.

**Οι βασικές λειτουργίες του GNUHealth είναι οι εξής:**

- Εξέταση ασθενούς και Λήψη ιστορικού
- Διαχείριση ασθενών (δημιουργία νέου ασθενούς, εκτίμηση, ιστορικού, κλπ)
- Διαχείριση ιατρών
- Διαχείριση Εργαστηρίου
- Πληροφορίες φαρμάκων
- Αποθήκη Ιατρο-νοσηλευτικού υλικού & Διαχείριση Προμηθειών
- Οικονομικο-Λογιστική διαχείριση Νοσοκομείου

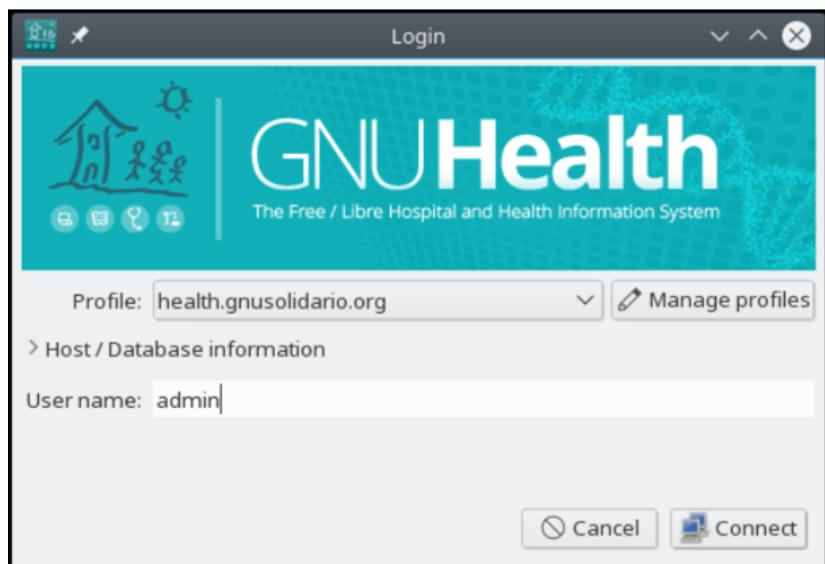


**Εικόνα 2.16:** Πλατφόρμες GNU Health [19]

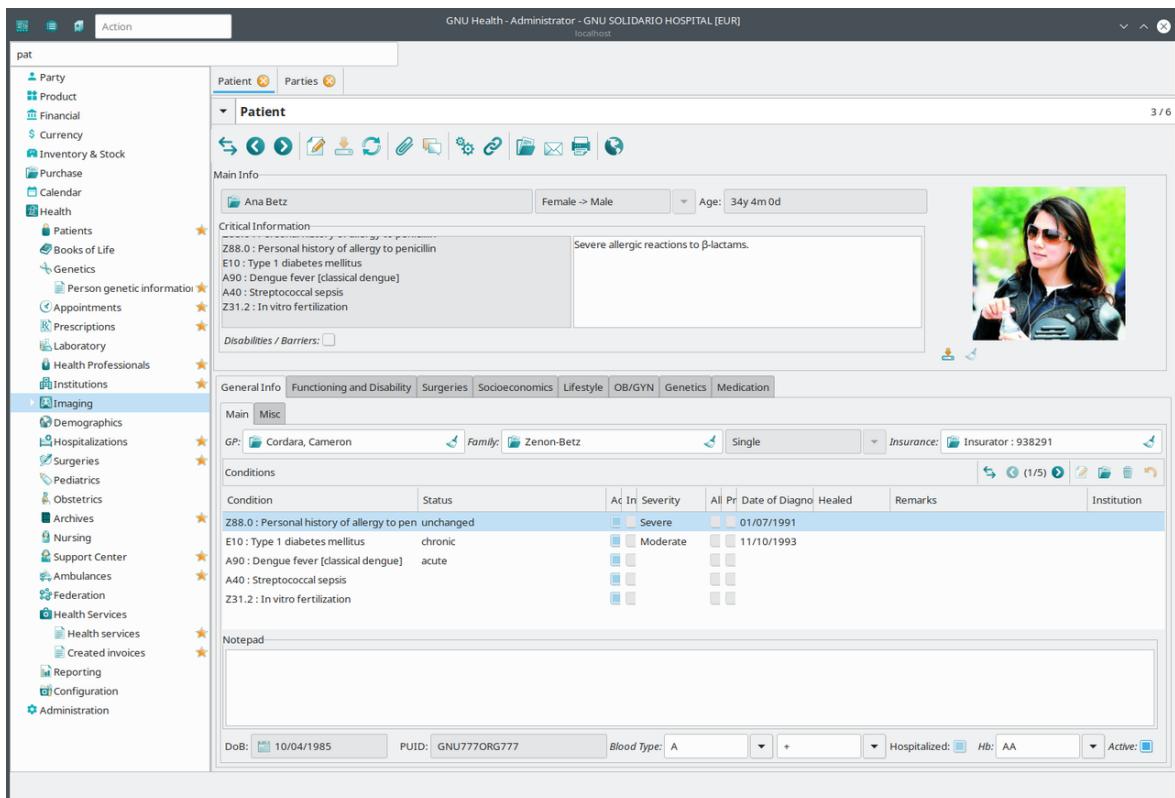
To GNU Health είναι ένα οικοσύστημα από ετερογενή αυτόνομα στοιχεία (που ονομάζονται πλατφόρμες):

- Το υποσύστημα διαχείρισης πληροφοριών για νοσοκομείο (Hospital Management Information System (H.M.I.S.))
- Το υποσύστημα διαχείρισης πληροφοριών εργαστηρίου (Laboratory Information System (L.I.M.S.) node).
- Έναν κατάλογο προσώπων (Person master index).
- Έναν γενικό εξυπηρετητή μηνυμάτων (Message server (Thalamus))
- Το πληροφοριακό υποσύστημα υγείας (Health Information System).

Το 2012 βραβεύτηκε ως το Καλύτερο Έργο Κοινωνικού Οφέλους από το Ιδρυμα Ελεύθερου Λογισμικού στο LibrePlanet στο Πανεπιστήμιο της Μασαχουσέτης της Βοστώνης. Το βραβείο FSF προωθεί τη χρήση Ελεύθερου Λογισμικού στην υπηρεσία της ανθρωπότητας.



**Εικόνα 2.17:** Οθόνη σύνδεσης [20]



Εικόνα 2.18: Αρχική Οθόνη ασθενούς στο Tryton [20]

Tryton					
Parties Patients					
Filters Search					
Patient	Lastname	SSN	ID	Hospitalization	
Betz, Ana	Betz	55567890	PAC001	hospitalized	
Zenón, John	Zenón	40556644	PAC002	outpatient	
Opoku, Lillian	Opoku	3434435	PAC003	outpatient	
Zenon Betz, Matt	Zenon Betz	97234436	PAC005	outpatient	
Kyalo, Dennis	Kyalo	54527954	PAC006	outpatient	
Doe, Juan	Doe	5555556	PAC008	outpatient	
Valério, Sara	Valério	147896547	PAC009	hospitalized	
López García, Horacio	López García	32436732	PAC010	outpatient	
Kowalski, Jan	Kowalski	3464565	PAC014	outpatient	
Lin, Jason	Lin	111444777	PAC015	outpatient	
Okinawa, Janna	Okinawa	9456789	PAC016	hospitalized	
seymore, jane	seymore	6344367	PAC019	outpatient	

Εικόνα 2.19: Λίστα Ασθενών [20]

### 2.6.3 OpenMRS



**Εικόνα 2.20:** Λογότυπο OpenMRS [21]

Η εφαρμογή OpenMRS είναι ένα ανοικτού κώδικα Ηλεκτρονικό σύστημα Ιατρικών Αρχείων (EMR) αρχικά σχεδιασμένο για αναπτυσσόμενες χώρες. Δημιουργήθηκε σε συνεργασία με την κοινότητα και λειτουργεί από το Φεβρουάριο του 2006 μέχρι και σήμερα. Χαρακτηρίζεται ταυτόχρονα, λογισμικό και κοινότητα και έχει εξελιχθεί σε μια πλατφόρμα ιατρικής πληροφορικής που χρησιμοποιείται πλέον σε κάθε ήπειρο.

Η εν λόγω εφαρμογή αρχικά κατασκευάστηκε για να υποστηρίξει τη φροντίδα του HIV/AIDS στην Κένυα και τη Ρουάντα. Ωστόσο, σχεδιάστηκε ως ένα γενικό σύστημα ιατρικών αρχείων που μπορεί να υποστηρίξει τη φροντίδα των ασθενών για διάφορες ιατρικές καταστάσεις, συμπεριλαμβανομένων της ελονοσίας, της φυματίωσης, αναγκών πρωτοβάθμιας φροντίδας και πολλά άλλα ζητήματα.

Ως εκ τούτου η εφαρμογή σχεδιάστηκε ως ένα γενικό σύστημα ιατρικών αρχείων που μπορεί να υποστηρίξει την φροντίδα ασθενών, να συλλέγει δεδομένα και παρατηρήσεις για κάθε ασθενή, να αποδίδει σε αυτά περιλήψεις, αναφορές και προβολές δεδομένων.

Το OpenMRS είναι ένα σύστημα ηλεκτρονικών ιατρικών αρχείων βασισμένο στη Java και στο διαδίκτυο, χρησιμοποιώντας την MySQL ως βάση δεδομένων. Ξεκίνησε με ένα απλό μοντέλο δεδομένων, το οποίο ενσωματώθηκε σε μια διεπαφή API, και στη συνέχεια δημιουργήθηκε μια εφαρμογή βασισμένη στο διαδίκτυο που χρησιμοποιεί αυτή την διεπαφή API.

Η εφαρμογή βασίζεται στις αρχές της διαφάνειας και της ανταλλαγής ιδεών, λογισμικού και στρατηγικών για ανάπτυξη και χρήση. Έχει σχεδιαστεί με την δυνατότητα να χρησιμοποιηθεί σε περιβάλλοντα με έλλειψη πόρων και μπορεί να τροποποιηθεί με την προσθήκη νέων δεδομένων, φορμών και εκθέσεων χωρίς προγραμματισμό. Γενικά αποτελεί μια πλατφόρμα που μπορεί να υιοθετηθεί από πολλούς οργανισμούς και δίνει το δικαίωμα για δωρεάν πρόσβαση στον πηγαίο κώδικα, επιτρέποντας τους, οποιαδήποτε τροποποίηση [22].

The screenshot shows the OpenMRS application interface. At the top, there is a navigation bar with links for Home, Find/Create Patient, Dictionary, Cohort Builder, Reporting, and Administration. The user is currently logged in as Super User. The main content area displays a patient profile for "John Doe" (Old Identification Number: 99999-9). Below the profile, there are tabs for Overview, Regimens, Encounters, Demographics, Graphs, and Form Entry. A "Patient Actions" section contains a button labeled "Exit Patient from Care". Under "Programs", it says "Not enrolled in any programs" and has a "Choose a program..." dropdown. The "Relationships" section lists a doctor named "Testarius Kungu Paul" and includes a link to "Add New Relationship". The "Allergies" section shows one entry for "GONORRHEA" on 18/11/2010 with a reaction of "SWELLING".

**Εικόνα 2.21:** Διεπαφή χρήστη της OpenMRS εφαρμογής [21]

The screenshot shows the OpenMRS application interface with a green header bar. The header includes the OpenMRS logo, the user name "admin", a location indicator for "Pharmacy", and a "Logout" button. Below the header, a green banner prompts the user to "Please tell us about your installation for the OpenMRS Atlas" and provides a "Configure Atlas" button. The main content area is titled "Logged in as Super User (admin) at Pharmacy." It features a grid of eight icons with labels: "Find Patient Record" (magnifying glass), "Active Visits" (calendar), "Register a patient" (person icon), "Capture Vitals" (heart rate monitor), "Appointment Scheduling" (calendar), "Reports" (document icon), "Data Management" (database icon), and "Configure Metadata" (cogwheel icon).

**Εικόνα 2.22:** Αρχική Οθόνη [21]

# Κεφάλαιο 3

## ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ

Στο παρόν κεφάλαιο θα παρουσιαστούν οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του προτεινόμενου συστήματος Ηλεκτρονικού Φακέλου Υγείας. Λόγω της πολυπλοκότητας και της μεγάλης κλίμακας του συστήματος, απαιτήθηκε η αξιοποίηση μιας ποικιλίας τεχνολογιών, καθεμία από τις οποίες έχει το δικό της ρόλο στην αρχιτεκτονική του συστήματος. Δίνεται ιδιαίτερη έμφαση στην ανάλυση των τεχνολογιών που αποτελούν τον ”σκελετό” της υλοποίησης και διατυπώνονται οι έννοιες που σχετίζονται με αυτές τις τεχνολογίες, καθώς και ο τρόπος με τον οποίο αυτές αλληλοεπιδρούν. Η ανάλυση καλύπτει τόσο τις τεχνολογίες που σχετίζονται με την αποθήκευση και επεξεργασία δεδομένων (Cassandra, Apache Spark), όσο και τα εργαλεία (Kubernetes, Helm charts) που χρησιμοποιήθηκαν για την ανάπτυξη των εφαρμογών (FastAPI, ReactJS).

### 3.1 Kubernetes



Εικόνα 3.1: Λογότυπο Kubernetes [23]

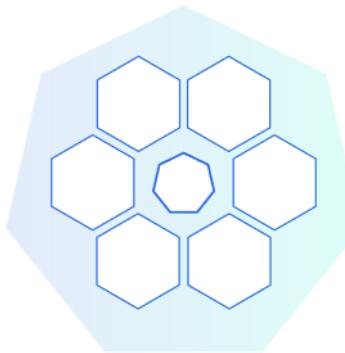
#### Τι είναι ο Κυβερνήτης (Kubernetes):

Ο Κυβερνήτης (Kubernetes) είναι ένα ανοιχτού κώδικα σύστημα ενορχήστρωσης που χρησιμοποιείται για την αυτοματοποίηση της εγκατάστασης, της κλιμάκωσης και της διαχείρισης εφαρμογών που εκτελούνται σε περιέκτες (containers). Αποσκοπεί στην παροχή ενός πλαισίου για την αυτόματη διαχείριση των υπηρεσιών (services) και των πόρων (resources) σε μεγάλης κλίμακας συστοιχίες (clusters), σε υπολογιστικά νέφη (cloud computing) κ.α., τα οποία μπορεί να αποτελούνται από εικονικές ή και φυσικές μηχανές. Είναι ιδιαίτερα αποτελεσματικό σε σενάρια όπου οι ανάγκες για δυναμική κλιμάκωση και υψηλή διαθεσιμότητα είναι κρίσιμες. Γενικά, το οικοσύστημα του Kubernetes προσφέρει έναν πλούσιο συνδυασμό

εργαλείων και συστατικών για τη διαχείριση σύνθετων, κατανεμημένων συστημάτων.

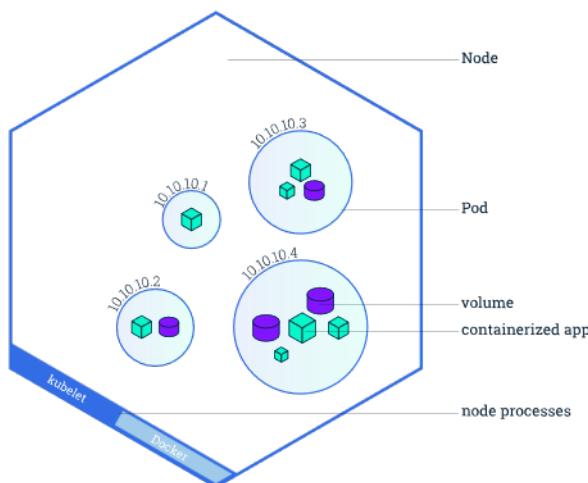
### **Βασικά Στοιχεία του Κυβερνήτη:**

- **Κόμβοι (Nodes):** Οι κόμβοι είναι οι φυσικοί ή εικονικοί υπολογιστές που συνθέτουν το cluster στο Kubernetes. Κάθε κόμβος εκτελεί ένα σύνολο υπηρεσιών, όπως τον kubelet τον kube-proxy καθώς και τα pods που ενσωματώνει. Οι κόμβοι μπορούν να είναι οποιουδήποτε τύπου, όπως φυσικοί υπολογιστές, εικονικές μηχανές ή συνδυασμό και των δύο. Τα nodes διακρίνονται σε δύο κατηγορίες:
  - **Master node:** Το master node είναι υπεύθυνο για τη διαχείριση του cluster.
  - **Worker nodes:** Οι worker nodes είναι υπεύθυνοι για την εκτέλεση των containers.



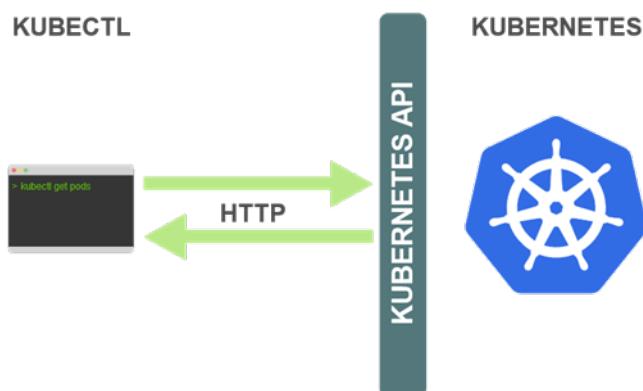
**Εικόνα 3.2:** Kubernetes cluster [23]

- **Pods:** Τα pods είναι οι βασικές εκτελεστικές μονάδες σε ένα cluster του Kubernetes. Κάθε pod φιλοξενεί ένα ή περισσότερα containers και διαχειρίζεται τη δικτύωση και τον αποθηκευτικό χώρο για αυτά τα containers. Τα pods είναι μικρά, αυτόνομα και μπορούν να δημιουργηθούν, να διαχειριστούν και να κλιμακωθούν μεμονωμένα, καθιστώντας τα εύκολα στη διαχείριση τους.
  - Τα containers είναι μικρές, ανεξάρτητες και αυτονομες μονάδες εκτέλεσης λογισμικού που περιλαμβάνουν όλα τα απαραίτητα για την εκτέλεση μιας εφαρμογής, όπως το λειτουργικό σύστημα, το λογισμικό και τα δεδομένα. Τα containers είναι ένα σημαντικό μέρος του Kubernetes, καθώς επιτρέπουν την απομόνωση και την εύκολη διαχείριση εφαρμογών.
- **kubeadm:** Το kubeadm είναι ένα εργαλείο που χρησιμοποιείται για τη δημιουργία ενός cluster στο Kubernetes. Το kubeadm αυτοματοποιεί τις βασικές εργασίες που απαιτούνται για τη δημιουργία ενός cluster, όπως η εγκατάσταση του Kubernetes σε κόμβους και την δημιουργία ενός δικτύου cluster.



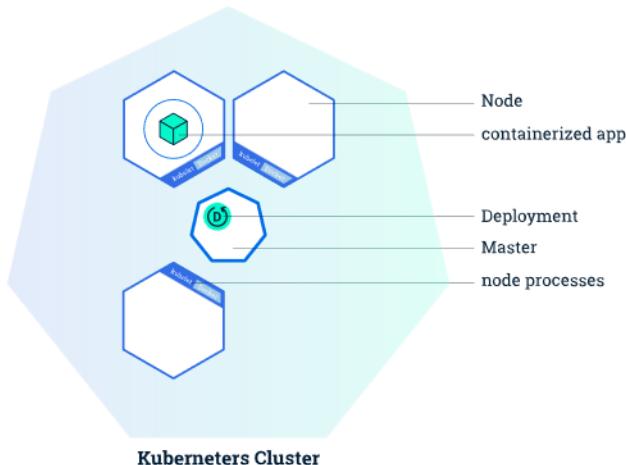
Εικόνα 3.3: Node [23]

- **kubectl:** Το kubectl είναι μια διεπαφή γραμμής εντολών ( command-line interface) που χρησιμοποιείται για την επικοινωνία με την διεπαφή (API) του Kubernetes. Το Kubernetes API χρησιμοποιείται για τη διαχείριση των clusters του Kubernetes. Με το kubectl, μπορούν να εκτελεστούν εντολές που δημιουργούν, διαχειρίζονται και καταργούν αντικείμενα στο Kubernetes, όπως pods, services και deployments.



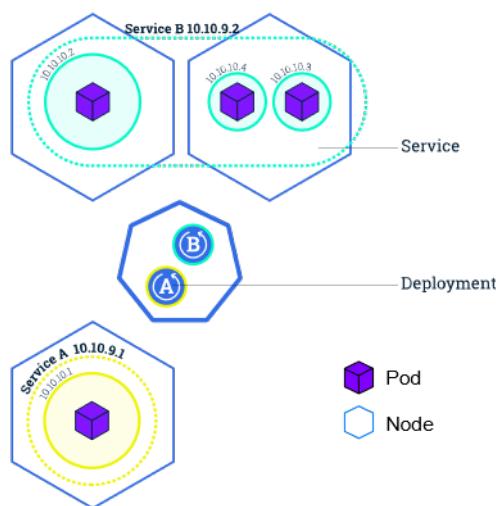
Εικόνα 3.4: kubectl [23]

- **Deployments:** Τα deployments είναι ένα είδος αντικειμένου Kubernetes που χρησιμοποιείται για την αυτοματοποίηση της δημιουργίας, της ενημέρωσης και της διαγραφής pods. Ένα deployment καθορίζει ένα σύνολο pods που εκτελούν την ίδια εφαρμογή. Τα deployments είναι μια αποτελεσματική μέθοδος για την κλιμάκωση και την ενημέρωση εφαρμογών σε ένα cluster του Kubernetes.



Εικόνα 3.5: Deployment [23]

- Services:** Οι υπηρεσίες είναι ένα σύνολο pods που είναι συνδεδεμένα μεταξύ τους μέσω κανόνων δικτύωσης. Οι υπηρεσίες παρέχουν ένα μοναδικό σημείο πρόσβασης σε ένα ή περισσότερα pods. Οι υπηρεσίες είναι απαραίτητες για την πρόσβαση σε pods από εξωτερικούς χρήστες και καθορίζουν τους κανόνες πρόσβασης τους.



Εικόνα 3.6: Services [23]

- ConfigMaps και Secrets:** Τα ConfigMaps και τα Secrets είναι δύο τύποι αντικειμένων του Kubernetes που χρησιμοποιούνται για την αποθήκευση δεδομένων. Τα ConfigMaps χρησιμοποιούνται για την αποθήκευση κανονικών δεδομένων, όπως ρυθμίσεις εφαρμογής. Τα Secrets χρησιμοποιούνται για την αποθήκευση ευαίσθητων δεδομένων, όπως ονόματα χρηστών και κωδικοί πρόσβασης. Τα ConfigMaps και τα Secrets είναι μια αποτελεσματική μέθοδος αποθήκευσης δεδομένων που μπορούν να χρησιμοποιηθούν από πολλά pods.
- Persistent Volumes (PVs):** Τα Persistent Volumes (PVs) είναι ένα είδος αντικειμένου Kubernetes που χρησιμοποιείται για την παροχή μόνιμης αποθήκευσης σε pods.

Τα PVs διαφέρουν από τα κανονικά volumes, καθώς παρέχουν μόνιμο αποθηκευτικό χώρο που διατηρείται ακόμα και αν το pod καταστραφεί ή διακοπεί. Τα PVs είναι μια αποτελεσματική μέθοδος για την αποθήκευση δεδομένων, καθιστώντας τα δεδομένα αυτά διαθέσιμα σε πολλά pods.

### 3.1.1 Helm Charts στο Kubernetes



**Εικόνα 3.7:** Λογότυπο Helm Charts [24]

Το Helm charts είναι ένα εργαλείο που χρησιμοποιείται για την αυτοματοποίηση της δημιουργίας, της διαχείρισης και της ενημέρωσης των εφαρμογών Kubernetes. Ένα Helm chart είναι ένα αρχείο με γλώσσα σειριοποίησης δεδομένων αναγνώσιμη από τον άνθρωπο (YAML) που περιγράφει ένα σύνολο αντικειμένων του Kubernetes, όπως pods, services και deployments. Το αρχείο αυτό μπορεί να χρησιμοποιηθεί για να δημιουργήσει, να διαχειριστεί και να ενημερώσει αυτά τα αντικείμενα. Είναι μια καλή επιλογή για την απλοποίηση της διαδικασίας εγκατάστασης και διαχείρισης εφαρμογών στο Kubernetes χρησιμοποιώντας ετοιμα templates και αποφεύγοντας την πολύπολη διαδικασία μέσω του ίδιου του Kubernetes.

Ένα Helm chart αποτελείται από τα εξής συστατικά:

- Values: Ένα σύνολο μεταβλητών που μπορούν να χρησιμοποιηθούν για την προσαρμογή της εφαρμογής
- Templates: Αρχεία που χρησιμοποιούνται για τη δημιουργία αντικειμένων Kubernetes
- Dependencies: Αναφορές σε άλλα Helm charts που απαιτούνται για την εγκατάσταση της εφαρμογής

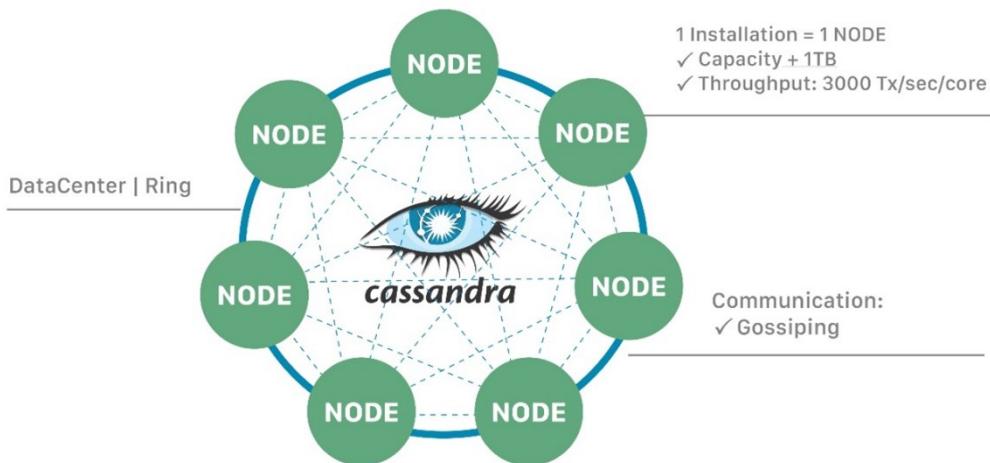
## 3.2 Apache Cassandra



**Εικόνα 3.8:** Λογότυπο Apache Cassandra [25]

Η Apache Cassandra είναι μια ανοιχτού κώδικα, κατανεμημένη, υψηλής διαθεσιμότητας, μη σχεσιακή (NoSQL) βάση δεδομένων. Είναι σχεδιασμένη να εκτελείται σε πολλούς διαφορετικούς υπολογιστές και να λειτουργεί ακατάπαυστα, ακόμη και αν κάποιοι από τους υπολογιστές στους οποίους εκτελείται καταστραφούν. Επίσης έχει την δυνατότητα να χειρίζεται μεγάλες ποσότητες δεδομένων, να παρέχει υψηλή διαθεσιμότητα χωρίς κανένα σημείο αποτυχίας και απώλειας δεδομένων.

**ApacheCassandra™ = NoSQL Distributed Database**

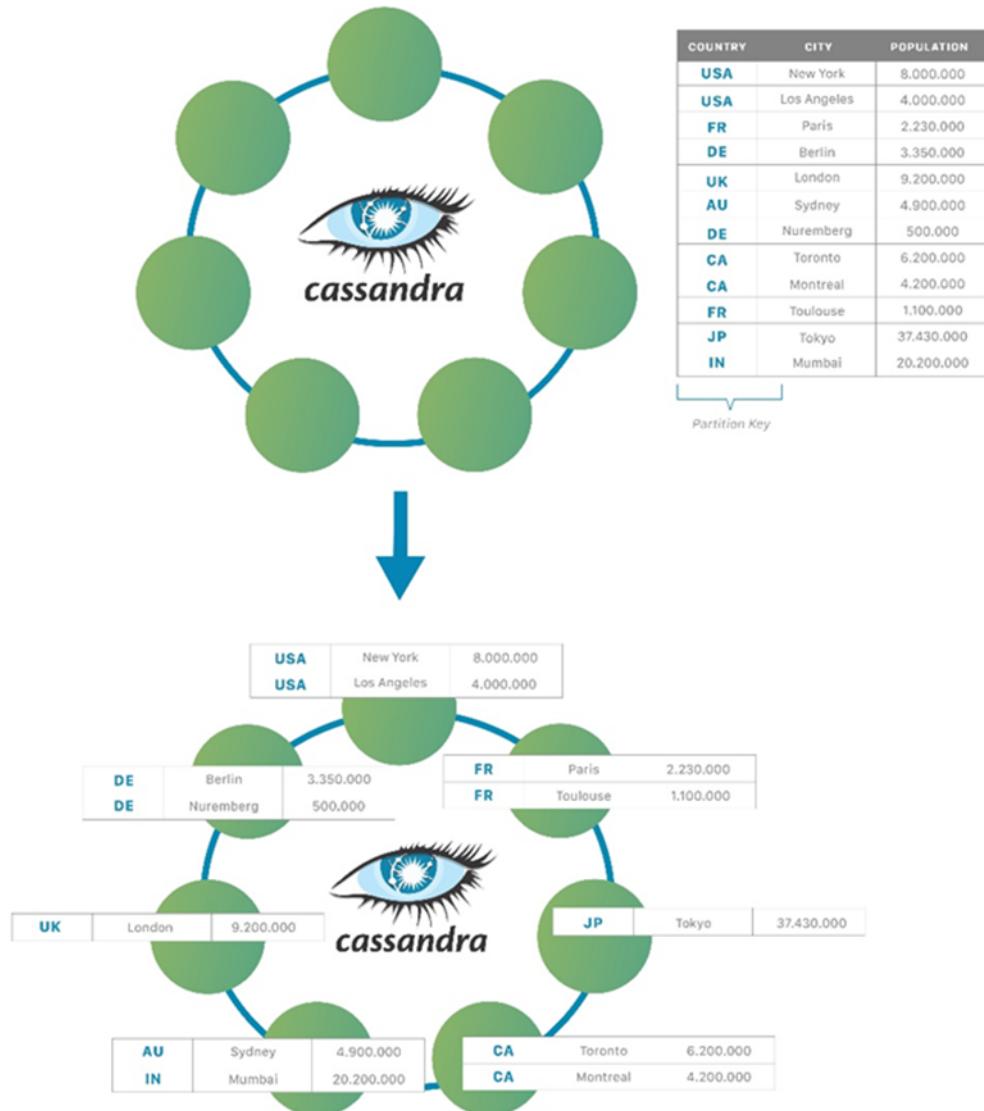


**Εικόνα 3.9:** Cassandra cluster [25]

Η Apache Cassandra αποτελείται από τα ακόλουθα μέρη:

- **Nodes:** Τα nodes είναι οι υπολογιστές στους οποίους εκτελείται η βάση δεδομένων. Τα nodes επικοινωνούν μεταξύ τους χρησιμοποιώντας το δίκτυο.
- **Cluster:** Ένα Cassandra cluster αποτελείται από δύο ή περισσότερα nodes. Είναι ένα σύνολο από nodes που συνεργάζονται για να παρέχουν την βάση δεδομένων της Cassandra.
- **Keyspace:** Το keyspace είναι μια συλλογή από tables και παρέχει ένα μοναδικό όνομα χώρου για τα δεδομένα που αποθηκεύονται στη βάση δεδομένων της Cassandra.

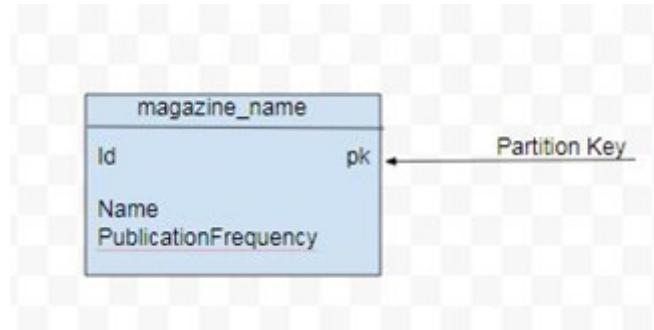
- **Table:** Ένα table είναι μια συλλογή από εγγραφες (rows) και χρησιμοποιείται για την αποθήκευση των δεδομένων.
- **Row:** Γραμμή (row) είναι μια συλλογή από στήλες (columns) και αντιπροσωπεύει μια μοναδική εγγραφή δεδομένων.
- **Column:** Ένα column είναι μια μονάδα δεδομένων και μπορεί να είναι οποιουδήποτε τύπου δεδομένων, όπως κείμενο, αριθμός, ημερομηνία ή ώρα.



**Εικόνα 3.10:** Αποθήκευση δεδομένων σε ένα Cassandra cluster [25]

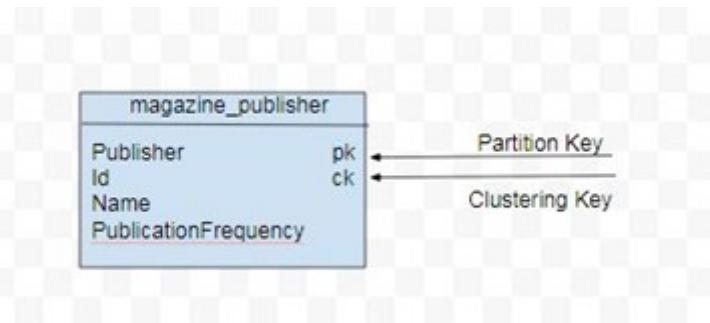
**Partition key:** Το partition key είναι ένα μοναδικό κλειδί που χρησιμοποιείται για την κατανομή των δεδομένων σε κάθε κόμβο. Κάθε εγγραφή σε μια βάση δεδομένων Cassandra έχει τουλάχιστον ένα partition key, το οποίο χρησιμοποιείται για την επιλογή του κόμβου στον οποίο θα αποθηκευτεί η εγγραφή. Οι εγγραφές με το ίδιο partition key θα αποθηκευτούν στο

ίδιο node. Αυτό επιτρέπει στην Cassandra να κατανέμει καλυτέρα τα δεδομένα σε όλο το cluster και να βελτιώσει την απόδοση των ερωτήσεων. Το partition key είναι υποχρεωτικό σε ένα table.



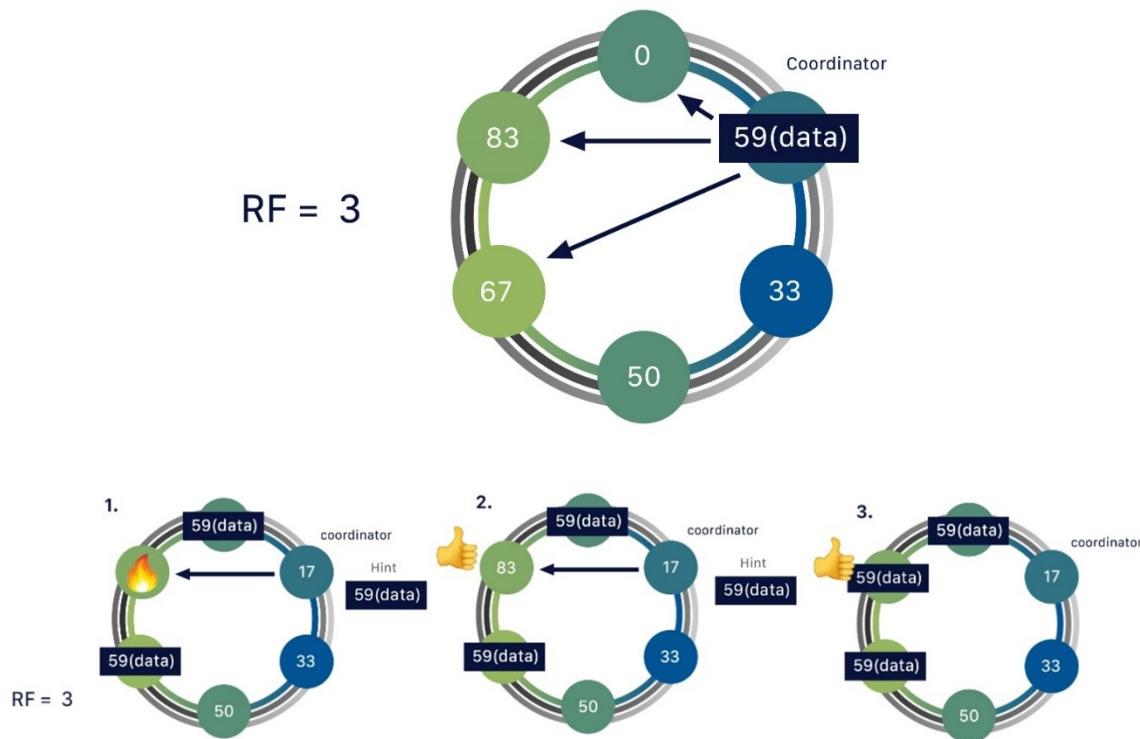
Εικόνα 3.11: Partition key [25]

**Cluster key:** Το cluster key είναι ένα πεδίο ή μια ομάδα πεδίων που χρησιμοποιείται για τη ταξινόμηση των δεδομένων σε ένα partition. Το cluster key δεν είναι απαραίτητο, αλλά μπορεί να χρησιμοποιηθεί για τη βελτίωση της απόδοσης των ερωτήσεων. Συγκεκριμένα το cluster key καθορίζει τη σειρά των εγγραφών σε ένα partition. Οι εγγραφές με το ίδιο partition key θα ταξινομηθούν με βάση το cluster key. Αυτό μπορεί να είναι χρήσιμο για ερωτήσεις που απαιτούν να επιστρέφονται οι εγγραφές με μια συγκεκριμένη σειρά.



Εικόνα 3.12: Cluster and Partition keys [25]

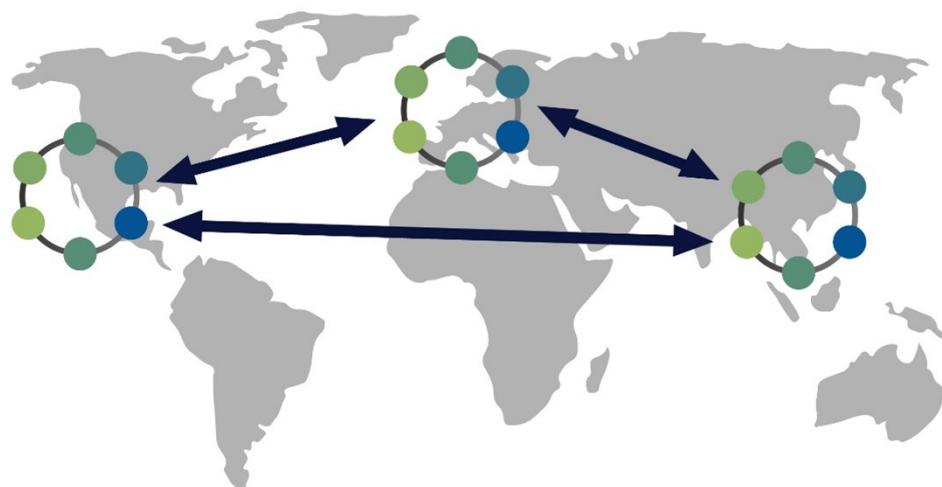
**Replication:** Η Apache Cassandra χρησιμοποιεί ένα μοντέλο αντιγραφής που ονομάζεται "replication factor". Το replication factor είναι ο αριθμός των nodes που αποθηκεύονται σε ένα αντίγραφο ενός partition. Για παράδειγμα, αν το replication factor είναι 3, τότε κάθε partition θα αποθηκεύεται σε τρία διαφορετικά nodes. Αυτό σημαίνει ότι ακόμα κι αν ένας node καταστραφεί, θα υπάρχουν ακόμα δύο αντίγραφα των δεδομένων στο cluster, ενώ με την αντικατάσταση του κατεστραμμένου αυτού node τα δεδομένα θα αντιγράφουν αυτόματα από το node που ακόμη τα περιέχει και θα επέλθει εκ νέου η ισορροπία των αντιγράφων.



Εικόνα 3.13: Replication Appache Cassandra [25]

**Topology:** Η αρχιτεκτονική ενός cluster Cassandra είναι γνωστή ως topology. Το topology καθορίζει πώς τα nodes είναι διατεταγμένα και πώς αλληλεπιδρούν μεταξύ τους. Η Apache Cassandra υποστηρίζει δύο τύπους topologies:

- Single datacenter: Όλα τα nodes βρίσκονται στο ίδιο data center.
- Multidatacenter: Τα nodes βρίσκονται σε δύο ή περισσότερα data centers.



Εικόνα 3.14: Multidatacenter in Cassandra [25]

Τέλος, η Apache Cassandra είναι μια βάση δεδομένων που χρησιμοποιεί τη γλώσσα ερωτημάτων CQL (Cassandra Query Language). Η CQL είναι παρόμοια με τη SQL, αλλά είναι προσαρμοσμένη για την αρχιτεκτονική και τις ιδιαιτερότητες της Cassandra. Ο συνδυασμός των κλειδιών, δηλαδή των Partitions keys και των Cluster Keys, είναι ζωτικής σημασίας για την αποτελεσματικότητα των ερωτημάτων στη CQL. Η σωστή επιλογή και διάταξη αυτών των κλειδιών καθορίζει την ταχύτητα και την αποδοτικότητα των ερωτημάτων, καθώς και τον τρόπο με τον οποίο τα δεδομένα οργανώνονται και αποθηκεύονται.

Η CQL και η επιλογή των κλειδιών είναι δύο βασικοί παράγοντες που συμβάλλουν στην αποδοτικότητα και την ευελιξία της Apache Cassandra.

### 3.3 Apache Spark



**Εικόνα 3.15:** Λογότυπο Apache Spark [26]

Το Apache Spark είναι ένα ανοιχτό κώδικα, κατανεμημένο σύστημα επεξεργασίας δεδομένων και ανάλυσης μεγάλου όγκου δεδομένων. Το Spark είναι ένα γρήγορο και ευέλικτο σύστημα που μπορεί να χρησιμοποιηθεί για μια ποικιλία εργασιών ανάλυσης, όπως η μηχανική μάθηση, η επεξεργασία φυσικής γλώσσας και η ανάλυση δεδομένων σε μεμονωμένα συστήματα ή και clusters. Η ικανότητα του να επιταχύνει διαδικασίες συνθέτων ερωτημάτων και αναλύσεων σε μεγάλου όγκου δεδομένα το καθιστά μια από τις πιο διαδομένες επιλογές στο τομέα του.

**To Apache Spark αποτελείται από τα ακόλουθα μέρη:**

1. Core: Το Core είναι το βασικό πλαίσιο του Spark. Το Core παρέχει τις βασικές λειτουργίες του Spark, όπως η διανομή δεδομένων, η διαχείριση πόρων και η εκτέλεση εργασιών.
2. SQL: Το SQL είναι ένα σύνολο εργαλείων και βιβλιοθηκών που επιτρέπουν στους χρήστες να εκτελούν ερωτήματα SQL σε δεδομένα που αποθηκεύονται στο Spark.
3. MLlib: Το MLlib είναι μια βιβλιοθήκη μηχανικής μάθησης που παρέχει μια ποικιλία αλγορίθμων μηχανικής μάθησης για την ανάλυση δεδομένων.
4. GraphX: Το GraphX είναι μια βιβλιοθήκη επεξεργασίας γραφημάτων που παρέχει λειτουργίες για την εργασία με δεδομένα και γραφήματα.

5. Spark Streaming: Το Spark Streaming είναι ένα πλαίσιο για την επεξεργασία δεδομένων ροής πραγματικού χρόνου.

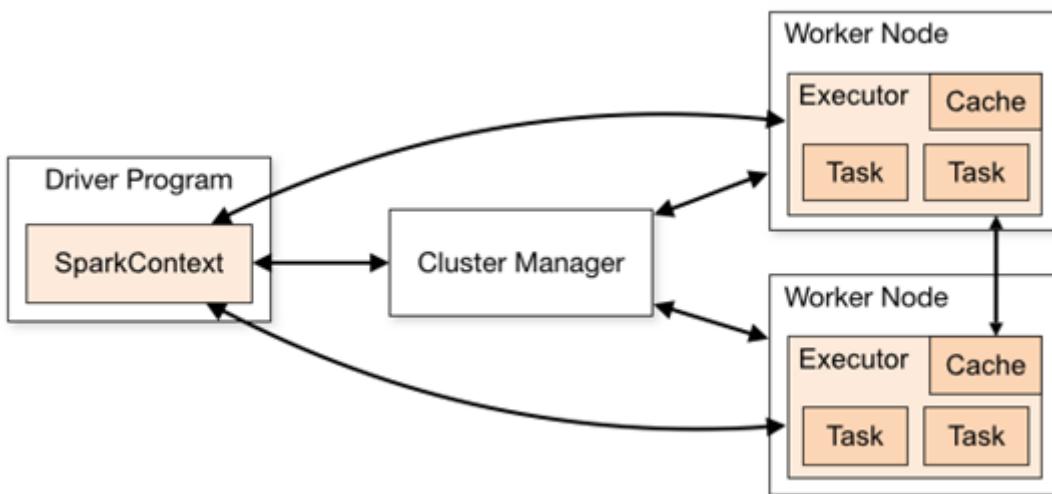
**Λειτουργία του Apache Spark:** Το Apache Spark λειτουργεί δημιουργώντας μια κατανεμημένη δομή δεδομένων που ονομάζεται Resilient Distributed Dataset (RDD). Ένα RDD είναι μια συλλογή από στοιχεία partitions. Συγκεκριμένα τα partitions είναι τα κομμάτια στα οποία χωρίζονται τα RDDs και κάθε partition είναι αποθηκευμένο σε έναν υπολογιστή.

Με τα partitions, το Spark μπορεί να επεξεργάζεται τα δεδομένα παράλληλα, γεγονός που μειώνει τον χρόνο εκτέλεσης και μεγιστοποιεί την απόδοση της επεξεργασίας.

Το Spark χρησιμοποιεί ένα μοντέλο επεξεργασίας δύο σταδίων για την ανάλυση δεδομένων. Στο πρώτο στάδιο, τα δεδομένα χωρίζονται σε μικρά partitions που μπορούν να επεξεργαστούν παράλληλα. Στο δεύτερο στάδιο, τα αποτελέσματα από τα διάφορα αυτά partitions συνδυάζονται για να δημιουργήσουν ένα τελικό αποτέλεσμα.

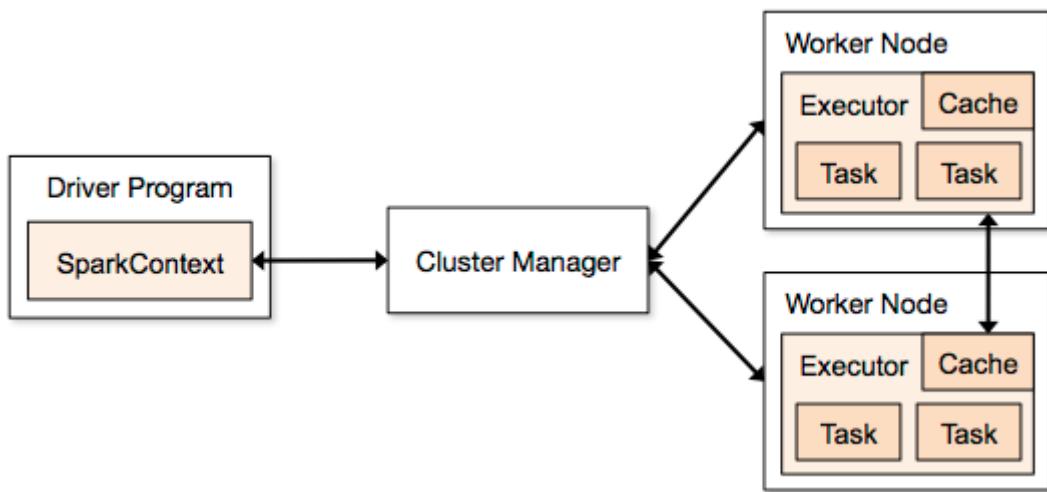
**Αρχιτεκτονικές Apache Spark:** Το Apache Spark υποστηρίζει τέσσερις αρχιτεκτονικές στην παρούσα έκδοση του:

- **Stand-alone:** Η αρχιτεκτονική stand-alone είναι η απλούστερη αρχιτεκτονική του Spark. Σε αυτή την αρχιτεκτονική, το Spark εκτελείται σε ένα cluster που αποτελείται από έναν master και έναν ή περισσότερους workers. Ο master διαχειρίζεται το cluster και κατανέμει τις εργασίες στους workers. Οι workers εκτελούν τις εργασίες και τα αποτελέσματα επιστρέφονται στον master.



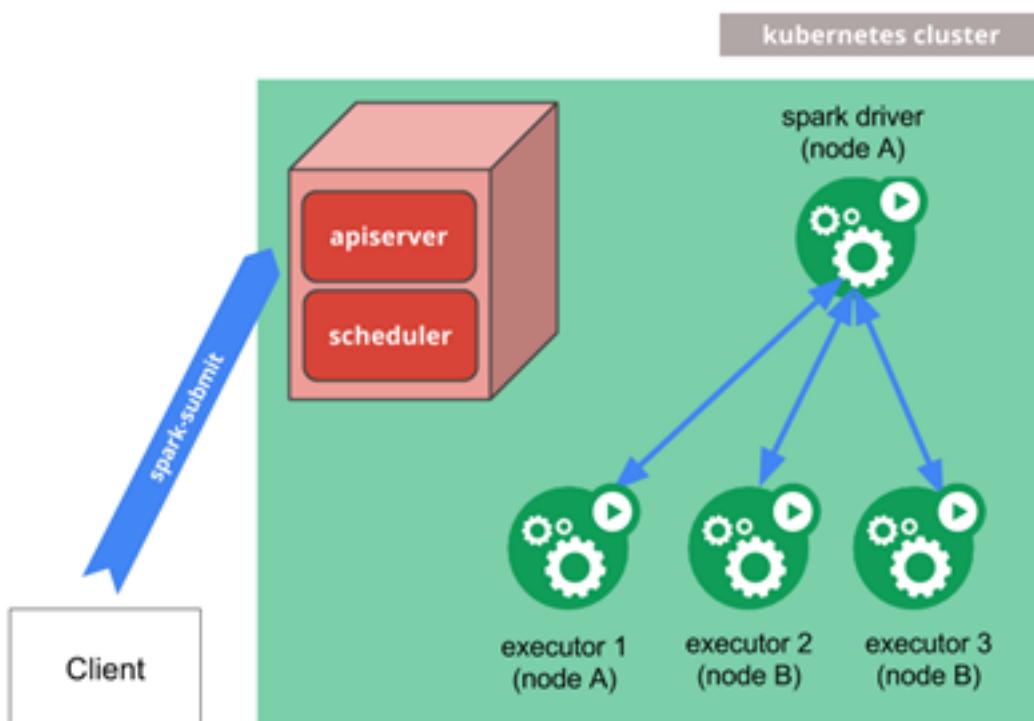
Εικόνα 3.16: Spark Stand-alone [26]

- **Mesos:** Το Mesos είναι ένα σύστημα διαχείρισης cluster που μπορεί να χρησιμοποιηθεί για την εκτέλεση πολλών διαφορετικών εφαρμογών, συμπεριλαμβανομένου του Spark. Στην αρχιτεκτονική Mesos, το Spark εκτελείται ως ένα Mesos framework. Το Mesos κατανέμει τις εργασίες του Spark στους workers του cluster.



Εικόνα 3.17: Spark Mesos [26]

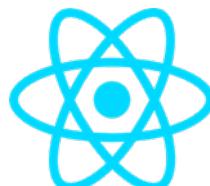
- **YARN:** Η YARN αρχιτεκτονική είναι μια ολοκληρωμένη αρχιτεκτονική που υποστηρίζεται από το Hadoop. Στην YARN αρχιτεκτονική, το Spark τρέχει ως application στο YARN cluster. Το YARN cluster αποτελείται από έναν resource manager, έναν application master και έναν ή περισσότερους workers. Το YARN κατανέμει τις εργασίες του Spark στους workers του cluster.
- **Kubernetes:** Σε αυτήν την αρχιτεκτονική το Spark εκτελείται ως ένα Kubernetes application. Με αυτόν τον τρόπο το Kubernetes κατανέμει τις εργασίες του Spark στους workers του cluster ή οποίοι παράγονται και διαχειρίζονται από το ίδιο το Kubernetes.



Εικόνα 3.18: Spark in Kubernetes [26]

Η επιλογή της κατάλληλης αρχιτεκτονικής για το Spark εξαρτάται από τις ανάγκες της εφαρμογής. Για απλές εφαρμογές, η αρχιτεκτονική stand-alone μπορεί να είναι επαρκής. Για εφαρμογές που απαιτούν μεγαλύτερη ευελιξία και δυνατότητα κλιμάκωσης, η αρχιτεκτονική Mesos, YARN ή Kubernetes μπορεί να είναι η καλύτερη επιλογή.

### 3.4 React JS

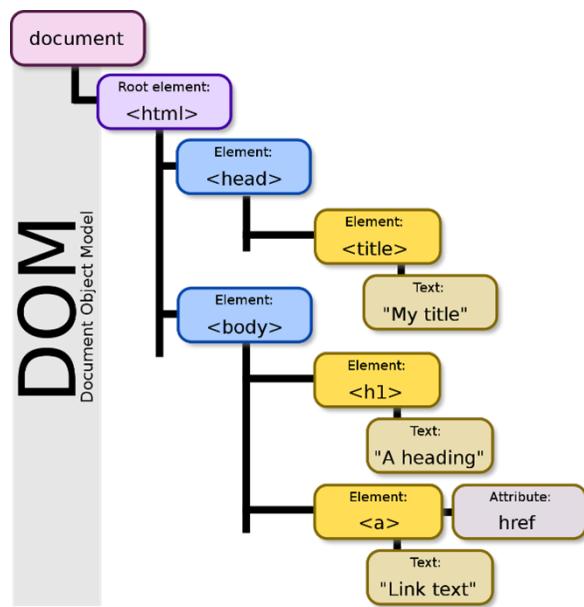


**Εικόνα 3.19:** Λογότυπο React JS [27]

Η React JS, είναι μια ανοιχτού κώδικα βιβλιοθήκη γραμμένη σε JavaScript που χρησιμοποιείται για την κατασκευή δυναμικών ιστοσελίδων και εφαρμογών. Βασίζεται στην ιδέα των στοιχείων (components), τα οποία είναι μικρές, επαναχρησιμοποιήσιμες μονάδες κώδικα javascript που μπορούν να συνδυαστούν για να δημιουργήσουν πιο σύνθετες σελίδες και εφαρμογές.

Η βιβλιοθήκη React παρέχει τα ακόλουθα βασικά στοιχεία:

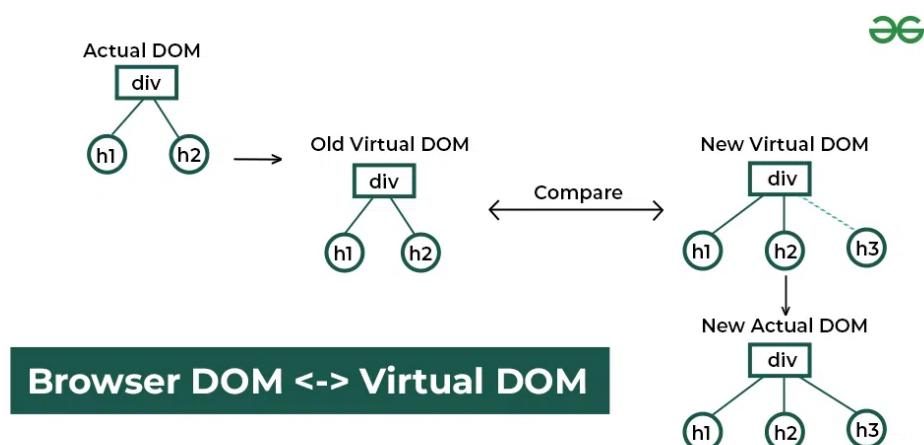
- Components:** Τα components είναι οι βασικές μονάδες κώδικα στοιχεία σε Javascript. Τα components είναι μικρά, επαναχρησιμοποιήσιμα κομμάτια κώδικα που μπορούν να συνδυαστούν για να δημιουργήσουν πιο σύνθετες σελίδες και εφαρμογές. Για παράδειγμα ένα στοιχείο θα μπορούσε να αποτελεί ένα κουμπί, μια φόρμα εισαγωγής είτε και κάτι πολύ πιο σύνθετο.
- State:** Το state είναι η κατάσταση ενός component. Το state μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων που αλλάζουν με την πάροδο του χρόνου.
- Props/Properties:** Τα props είναι οι παραμέτροι που μπορούν να περάσουν σε ένα component.
- DOM:** Το DOM (Document Object Model) είναι ένα γραφικό μοντέλο που αντιπροσωπεύει τη δομή μιας ιστοσελίδας ή εφαρμογής.
- Virtual DOM:** Το virtual DOM είναι μια ελαφριά, ψηφιακή αναπαράσταση του πραγματικού DOM. Το virtual DOM είναι μια πολύ απλή δομή που μπορεί να ενημερωθεί γρήγορα και αποτελεσματικά.



Εικόνα 3.20: DOM [28]

### Ο τρόπος λειτουργίας της React:

Η λειτουργία της React JS βασίζεται στο virtual DOM. Πιο συγκεκριμένα όταν συμβεί μια αλλαγή στο state της εφαρμογής, συγκρίνεται το virtual DOM με το πραγματικό DOM, έπειτα εντοπίζονται οι διαφορές μεταξύ τους και στη συνέχεια ενημερώνονται μόνο τα μέρη του πραγματικού DOM που έχουν αλλάξει. Αυτή η διαδικασία επιτρέπει στο να είναι πολύ πιο αποδοτική και γρήγορη η ενημέρωση της εμφάνισης και του περιεχόμενου της εφαρμογής από αντίστοιχες τεχνολογίες που λειτουργούν άμεσα στο DOM.



Εικόνα 3.21: Virtual Dom [29]

### 3.5 FastAPI



**Εικόνα 3.22:** Λογότυπο FastAPI [30]

To FastAPI είναι ένα ανοιχτού κώδικα, γρήγορο και ευέλικτο Python framework για την ανάπτυξη διεπεφών REST API. To FastAPI είναι ένα σχετικά νέο framework, παρ' όλα αυτά έχει γίνει γρήγορα δημοφιλή χάρη στην απλότητά, την απόδοσή και την ευκολία χρήσης του.

- To REST API (Representational State Transfer Application Programming Interface) είναι ένα πρότυπο αρχιτεκτονικής για την ανάπτυξη εφαρμογών web. Αποτελείται από ένα σύνολο κανόνων με τους οποίους καθορίζεται η επικοινωνία μεταξύ των υπολογιστών μέσω του Διαδικτύου. Ta REST API χρησιμοποιούνται συχνά για την παροχή πρόσβασης σε δεδομένα ή την επικοινωνία μεταξύ συστημάτων [31].
- Ta REST API χρησιμοποιούν τέσσερις βασικούς τύπους αιτημάτων για την επικοινωνία μεταξύ πελατών (clients) και διακομιστών (servers):
  - GET: To αίτημα GET χρησιμοποιείται για την ανάκτηση δεδομένων.
  - POST: To αίτημα POST χρησιμοποιείται για την εισαγωγή νέων δεδομένων.
  - PUT: To αίτημα PUT χρησιμοποιείται για την ενημέρωση υφιστάμενων δεδομένων.
  - DELETE: To αίτημα DELETE χρησιμοποιείται για τη διαγραφή δεδομένων.
- Ta REST API χρησιμοποιούν διάφορους τύπους δεδομένων κατά την επικοινωνία μεταξύ πελατών και διακομιστών. Oi πιο συνηθισμένοι τύποι δεδομένων είναι:
  - JSON: To JSON είναι ένας δημοφιλής τύπος δεδομένων για την επικοινωνία δεδομένων μεταξύ πελατών και διακομιστών.
  - XML: To XML είναι ένας άλλος δημοφιλής τύπος δεδομένων για την επικοινωνία δεδομένων μεταξύ πελατών και διακομιστών.
  - HTML: To HTML χρησιμοποιείται για την επικοινωνία δεδομένων μεταξύ πελατών και διακομιστών που είναι προσανατολισμένοι στην πλοήγηση.

### Ενδεικτικά κάποια από τα πιο βασικά συστατικά του FastAPI είναι:

- **Router:** Ένα από τα κύρια συστατικά, ο Router (δρομολογητής) βοηθά να οριστούν διάφορες διαδρομές API για την εφαρμογή (endpoint). Μπορεί να οριστούν οποιαδήποτε από τις τέσσερις βασικούς τύπους αιτημάτων (GET,PUT,POST,DELETE) σε διαφορετικές συναρτήσεις.
- **Παράμετροι διαδρομών και παράμετροι ερωτημάτων:** Το FastAPI επιτρέπει την εύκολη προσθήκη παραμέτρων μέσα από τη διαδρομή URL αλλά και τις παραμέτρους ερωτημάτων.
- **Request Body:** Επιτρέπει την αποστολή και λήψη διάφορων τύπων δεδομένων όπως JSON μέσα στο σώμα του αιτήματος HTTP και μπορεί να το μετατρέψει σε αντίστοιχες δομές δεδομένων Python όπως pandas dataframes κ.α .
- **Μοντέλα δεδομένων:** Το FastAPI χρησιμοποιεί μοντέλα Pydantic για τη μορφοποίηση αιτημάτων και απαντήσεων. Τα μοντέλα αυτά εκτελούν επικύρωση δεδομένων, σειριοποίηση και τεκμηρίωση (documentation).

### Επιπλέον συστατικά του είναι:

- **Authentication:** Το FastAPI παρέχει πολλαπλούς τρόπους για την εφαρμογή ελέγχου ταυτότητας, όπως OAuth2, JWT, και βασικός έλεγχος ταυτότητας HTTP.
- **Automatic Documentation:** Διαθέτει ενσωματωμένη μια αυτοματοποιημένη δημιουργία δια-δραστικών κείμενων τεκμηρίωσης (documentations) του API χρησιμοποιώντας εργαλεία όπως το Swagger UI.
- **Asynchronous Support:** Υποστηρίζει εγγενώς τον ασύγχρονο χειρισμό αιτημάτων, καθιστώντας το αποδοτικό για εργασίες με δεδομένα εισόδου-εξόδου (I/O).
- **CORS (Cross-Origin Resource Sharing):** Παρέχει ενσωματωμένα εργαλεία για την κοινοποίηση πόρων μεταξύ καταγωγής (CORS), επιτρέποντας στο frontend και το backend να επικοινωνούν με ασφάλεια.
- **Error Handling:** Παρέχει έναν απλό τρόπο χειρισμού εξαιρέσεων και επιστροφής προσαρμοσμένων απαντήσεων σφαλμάτων.

Τα χαρακτηριστικά του FastAPI το καθιστούν μια ευέλικτη επιλογή για τη δημιουργία API, από απλά έργα έως σύνθετες εφαρμογές με ποικίλες ανάγκες για επικύρωση δεδομένων, έλεγχο ταυτότητας και ασύγχρονο χειρισμό.

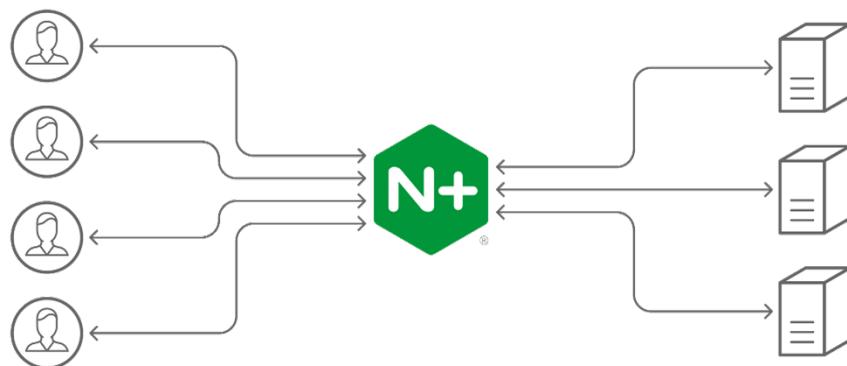
## 3.6 Nginx



**Εικόνα 3.23:** Λογότυπο Nginx [32]

Το NGINX είναι ένας ανοιχτού κώδικα διακομιστής web, που χρησιμοποιείται για διάφορες λειτουργίες, όπως αντίστροφος διακομιστής μεσολάβησης (reverse proxy), εξισορροπητής φορτίου (load balancer), διακομιστής μεσολάβησης (proxy server), διακομιστής κρυπτογράφησης HTTPS, και πολλά άλλα.

Το NGINX είναι ένα πολύ δημοφιλές λογισμικό που χρησιμοποιείται από πολλές εταιρίες ανά το κόσμο και είναι γνωστό για την υψηλή απόδοσή του, την αξιοπιστία του και την ευκολία χρήσης του.



**Εικόνα 3.24:** Nginx Load Balancer [32]

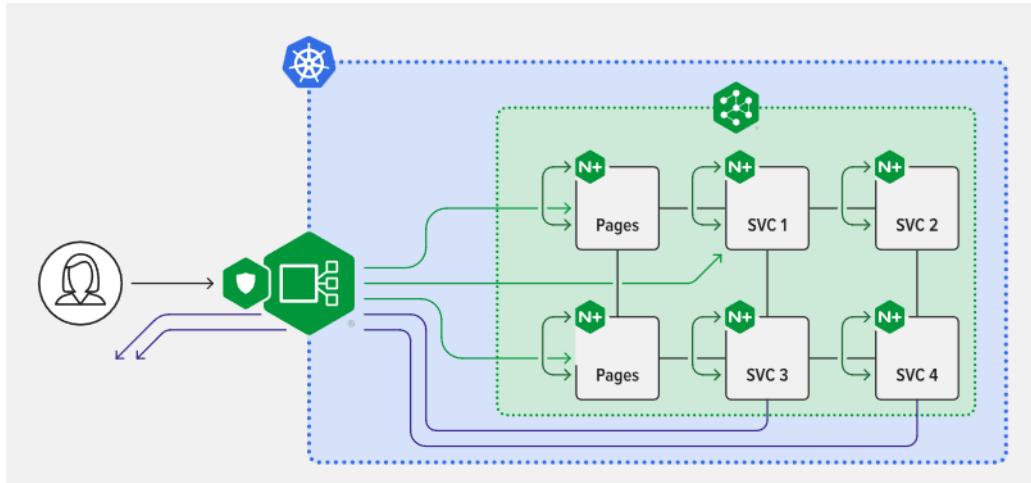
Αναλυτικά κάποιες από τις λειτουργίες του :

- Reverse proxy: Λαμβάνει τις εισερχόμενες συνδέσεις και τις ανακατευθύνει στον κατάλληλο διακομιστή, βελτιστοποιώντας την απόδοση.
- Load Balancer: Μπορεί να χρησιμοποιηθεί για την κατανομή αιτημάτων σε πολλούς διακομιστές, βελτιώνοντας την απόδοση και την αξιοπιστία.
- Proxy server: Μπορεί να χρησιμοποιηθεί ως ενδιάμεσος μεταξύ ενός χρήστη και ενός άλλου διακομιστή κτλ.

Το NGINX χαρακτηρίζεται από:

- Υψηλή απόδοση: είναι γνωστό για την υψηλή του απόδοση, ακόμη και σε μεγάλα φορτία.

- Συνδέσεις υψηλής ταχύτητας: Το NGINX μπορεί να χειρίστει αποτελεσματικά συνδέσεις υψηλής ταχύτητας, όπως αυτές που απαιτούνται για την υποστήριξη βίντεο και streaming και άλλων απαιτητικών εφαρμογών Web.
- Ασφάλεια: Το NGINX διαθέτει μια σειρά από χαρακτηριστικά ασφαλείας που μπορούν να βοηθήσουν στην προστασία του συστήματος.
- Ευελιξία: Το NGINX χρησιμοποιεί αρχεία διαμόρφωσης και ρυθμίσεων που καθορίζουν τον τρόπο της λειτουργίας του, καθιστώντας το εξαιρετικά ευέλικτο.
- Επεκτασιμότητα: Είναι σχεδιασμένο για να αντεπεξέλθει σε υψηλό φόρτο εργασίας, κάνοντάς το ιδανικό για συστήματα που χρειάζονται υψηλή επεκτασιμότητα.
- Το NGINX είναι πλήρως συμβατό με το Kubernetes και ένα εξαιρετικό εργαλείο για την επέκταση των δυνατοτήτων του



**Εικόνα 3.25:** Nginx in Kubernetes [32]

## Κεφάλαιο 4

# ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΗΦΥ

Στο παρόν κεφάλαιο παρουσιάζεται η αρχιτεκτονική του συστήματος, η οποία αποτελεί την καρδιά του ΗΦΥ που έχει αναπτυχθεί. Οι τεχνολογίες που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, οι λόγοι για την επιλογή τους και ο τρόπος που συνδυάζονται για την κατασκευή του συστήματος, θα γίνουν πλήρως κατανοητοί μέσα από την παρουσίαση της αρχιτεκτονικής.

### 4.1 Στόχοι και Λειτουργίες του Συστήματος με βάση την Αρχιτεκτονική

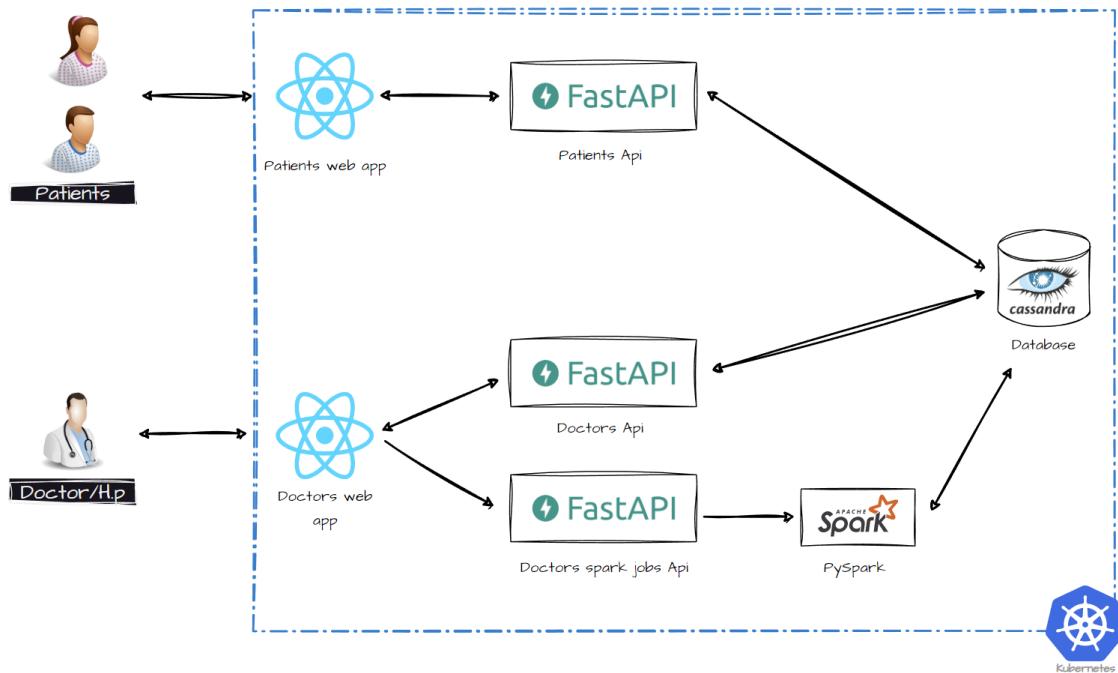
Ο στόχος της προτεινόμενης αρχιτεκτονικής είναι η υλοποίηση ενός ευέλικτου, ασφαλούς, και αξιόπιστου συστήματος διαχείρισης ηλεκτρονικού φακέλου υγείας. Η εφαρμογή του προτεινόμενου συστήματος θα καταστήσει δυνατή την ολοκληρωμένη και αποτελεσματική διαχείριση ιατρικών δεδομένων, θα διευκολύνει τον γρήγορο και ασφαλή διαμοιρασμό πληροφοριών μεταξύ των μονάδων υγείας, θα επιτρέψει τη λεπτομερή ανάλυση των δεδομένων για κλινικές αξιολογήσεις και θεραπευτικές αποφάσεις/αγωγές, και θα δώσει τη δυνατότητα στους ασθενείς για άμεση και ασφαλή πρόσβαση στα προσωπικά τους ιατρικά αρχεία. Με αυτόν τον τρόπο, το σύστημα θα αξιοποιεί πλήρως τα δεδομένα για το καλό των ασθενών, των επαγγελματιών υγείας και του ευρύτερου συστήματος υγείας.

Για να επιτευχθεί αυτός ο στόχος, το σύστημα πρέπει να πληροί τα ακόλουθα κριτήρια:

- Ευελιξία:** Το σύστημα πρέπει να είναι ευέλικτο ώστε να μπορεί να προσαρμοστεί στις μεταβαλλόμενες ανάγκες των χρηστών και του τομέα της υγείας.
- Ασφάλεια:** Το σύστημα πρέπει να είναι ασφαλές ώστε να προστατεύει τα ευαίσθητα δεδομένα των ασθενών.

- **Αξιοπιστία:** Το σύστημα πρέπει να είναι αξιόπιστο ώστε να μπορεί να παρέχει συνεχή πρόσβαση στα δεδομένα των ασθενών.
- **Κλίμακωση:** Το σύστημα πρέπει να είναι σε θέση να κλιμακωθεί για να υποστηρίξει τις ανάγκες ενός μεγάλου αριθμού ασθενών και γιατρών.
- **Παραλληλοποίηση:** Το σύστημα πρέπει να χρησιμοποιεί παράλληλα υπολογιστικά συστήματα για να βελτιώσει την απόδοσή του.
- **Κατανεμημένο:** Το σύστημα πρέπει να είναι κατανεμημένο σε πολλούς υπολογιστές για να αυξήσει την αξιοπιστία του καθώς και την απόδοση του.
- **Χρηστικότητα (Usability):** Η διεπαφή χρήστη (User Interface) του συστήματος πρέπει να είναι φιλική προς τον χρήστη, ενώ παράλληλα να προσφέρει προηγμένες λειτουργίες για επαγγελματίες το τομέα της υγείας.

Προκειμένου να υλοποιηθούνται τα παραπάνω κριτήρια η γενική ιδέα της αρχιτεκτονικής που έχει επιλεχθεί για το συγκεκριμένο σύστημα είναι όπως παρουσιάζεται στο παρακάτω διάγραμμα.



**Εικόνα 4.1:** Διάγραμμα Αρχιτεκτονικής Συστήματος

Το σύστημα που απεικονίζεται στο διάγραμμα αποτελείται από δύο εφαρμογές διαδικτύου (web apps) σε React JS, μία για τους ασθενείς και μία για τους επαγγελματίες υγείας. Κάθε εφαρμογή έχει το δικό της πίσω μέρος (backend) υλοποιημένο σε FastAPI, το οποίο επικοινωνεί με μια ενιαία βάση δεδομένων την Apache Cassandra. Για τους επαγγελματίες

υγείας, υπάρχει επίσης ένα επιπλέον backend που συνδέεται με το front end και είναι υπεύθυνο για την επικοινωνία με τον Spark. Ο Spark εκτελεί εργασίες ανάλυσης δεδομένων, τα αποτελέσματα των οποίων αποθηκεύονται στη βάση δεδομένων Apache Cassandra. Το σύνολο του συστήματος έχει αναπτυχθεί/εγκατασταθεί σε cluster στο Kubernetes.

## 4.2 Σενάρια χρήσης

Το σύστημα που κατασκευάστηκε υποστηρίζει δυο κατηγορίες χρηστών. Για κάθε μια κατηγορία χρήστη θα γίνει ανάλυση των σεναρίων χρήσης, τα οποία χρησιμοποιήθηκαν για τον σχεδιασμό του συστήματος.

### A) Σύστημα ασθενών

Το σύστημα ασθενών παρέχει μια σειρά από δυνατότητες που επιτρέπουν στους ασθενείς να παρακολουθούν την υγεία τους την πρόοδό τους και να διασφαλίζουν ότι λαμβάνουν την απαραίτητη φροντίδα. Τα ακόλουθα είναι τα βασικά σενάρια χρήσης (use cases) του συστήματος ασθενών:

- Εγγραφή και σύνδεση:** Οι ασθενείς μπορούν να δημιουργήσουν έναν λογαριασμό στο σύστημα και να συνδεθούν με τους κωδικούς πρόσβασής τους.
- Προσωπικά δεδομένα:** Οι ασθενείς μπορούν να βλέπουν και να επεξεργάζονται τα προσωπικά τους δεδομένα, όπως το όνομα, την ηλικία, την ημερομηνία γέννησης, τη διεύθυνση κατοικίας και το τηλέφωνο επικοινωνίας.
- Λήψη ειδοποίησεων:** Οι ασθενείς μπορούν να λαμβάνουν ειδοποίησεις για σφάλματα καταχώρησης ή τροποποιήσεις δεδομένων ή για αποτυχία λειτουργίας τμημάτων του συστήματος.
- Παρακολούθηση ιατρικών συνταγών:** Οι ασθενείς μπορούν να βλέπουν τις ιατρικές τους συνταγές και τις ημερομηνίες συνταγογράφησης.
- Παρακολούθηση επισκέψεων:** Οι ασθενείς μπορούν να βλέπουν μια λίστα με όλες τις επισκέψεις τους στους γιατρούς και τις μονάδες υγείας.
- Αναλυτική περιγραφή επισκέψεων:** Οι ασθενείς μπορούν να βλέπουν λεπτομερείς πληροφορίες για κάθε επίσκεψη τους, όπως το είδος της επίσκεψης, τη διάγνωση, τη συνταγογράφηση και τις παρατηρήσεις.
- Καθορισμός διαμοιρασμού επισκέψεων:** Ο ασθενής μπορεί να ορίσει για κάθε επίσκεψή του αν αυτή θα διαμοιραστεί με άλλους γιατρούς/επαγγελματίες υγείας ή όχι. Ο ασθενής θα μπορεί να δει όλες τις επισκέψεις του και για κάθε επίσκεψη, θα μπορεί

να ορίσει αν αυτή θα είναι ορατή στους άλλους γιατρούς ή όχι, εκτός του γιατρού/επαγγελματία υγείας που την καταχώρησε.

### B) Σύστημα επαγγελματιών υγείας

Τα βασικά σενάρια χρήσης του συστήματος επαγγελματιών υγείας είναι τα εξής:

- **Εγγραφή και σύνδεση:** Οι επαγγελματίες υγείας μπορούν να εγγραφούν στο σύστημα και να συνδεθούν με τους κωδικούς πρόσβασής τους.
- **Διαχείριση ασθενών:** Οι επαγγελματίες υγείας μπορούν να διαχειρίζονται τους ασθενείς τους, συμπεριλαμβανομένης της εγγραφής νέων ασθενών, της αναζήτησης ασθενών και της εμφάνισης των πληροφοριών τους.
- **Δημιουργία επισκέψεων:** Οι επαγγελματίες υγείας μπορούν να δημιουργούν επισκέψεις για τους ασθενείς τους, καταγράφοντας πληροφορίες όπως τον τίτλο, την ημερομηνία, τον τύπο, τη διάγνωση, τα συμπτώματα, τις ασθένειες, τη συνταγογράφηση και τις παρατηρήσεις.
- **Αναζήτηση επισκέψεων:** Οι επαγγελματίες υγείας μπορούν να αναζητούν επισκέψεις για τους ασθενείς τους. Ο επαγγελματίας υγείας μπορεί να δει επισκέψεις που έχουν διαμοιραστεί από τον ασθενή και αφορούν άλλους επαγγελματίες υγείας. Οι επισκέψεις αυτές εμφανίζονται ξεχωριστά από τις επισκέψεις που έχει δημιουργήσει ο ίδιος ο επαγγελματίας υγείας.
- **Πρόσβαση σε επισκέψεις άλλων επαγγελματιών υγείας:** Οι επαγγελματίες υγείας μπορούν να δουν επισκέψεις που έχουν δημιουργήσει άλλοι επαγγελματίες υγείας για έναν ασθενή που έχουν εγγράψει στη λίστα ασθενών τους μόνο αν ο ασθενής έχει ορίσει αυτές τις επισκέψεις ως διαμοιραζόμενες.
- **Πρόσβαση σε αναλύσεις δεδομένων υγείας για ασθενή:** Οι επαγγελματίας υγείας μπορούν να έχουν πρόσβαση σε αναλύσεις δεδομένων υγείας για έναν ασθενή. Οι αναλύσεις αυτές δημιουργούνται με βάση τις πληροφορίες που έχουν καταχωρηθεί στο σύστημα για τον ασθενή, συμπεριλαμβανομένων των επισκέψεων που έχουν δημιουργήσει οι επαγγελματίες υγείας και των επισκέψεων που έχουν διαμοιραστεί από τον ασθενή. Αυτές οι αναλύσεις παρέχουν πληροφορίες σχετικά με την υγεία του ασθενή, όπως η πορεία της νόσου, οι κίνδυνοι και οι δυνατότητες για βελτίωση.

## 4.3 Βάση δεδομένων Apache Cassandra

### 4.3.1 Επιλογή βάσης δεδομένων

Τα σενάρια χρήσης που περιεγράφηκαν στο προηγούμενο υπό-κεφάλαιο απαιτούν μια βάση δεδομένων που μπορεί να χειρίστει μεγάλες ποσότητες δεδομένων και να παρέχει γρήγορες απαντήσεις σε ερωτήματα, ειδικά όταν ο αριθμός των ασθενών και των επαγγελματιών υγείας που θα πρέπει να υποστηρίξει το σύστημα αυξηθεί κατακόρυφα. Η Cassandra αποτελεί μια καλή επιλογή για αυτές τις εφαρμογές, λόγω των ιδιαίτερων χαρακτηριστικών της (αναλύονται στο υπό κεφάλαιο 3.2). Κάποια από τα βασικά χαρακτηριστικά της είναι:

- **Ανθεκτική:** Η Cassandra είναι σχεδιασμένη για να είναι ανθεκτική σε αστοχίες κόμβων, ώστε να μπορεί να συνεχίσει να λειτουργεί ακόμη και αν ένας ή περισσότεροι κόμβοι καταστραφούν ή αποτύχουν. Αυτό είναι σημαντικό για εφαρμογές υγείας, οι οποίες πρέπει να είναι διαθέσιμες 24/7.
- **Κλιμακούμενη:** Η Cassandra μπορεί να κλιμακωθεί προσθέτοντας περισσότερους κόμβους στο σύστημα. Αυτό επιτρέπει στο σύστημα να υποστηρίξει περισσότερους χρήστες και δεδομένα χωρίς να επηρεάζεται η απόδοση.
- **Μπορεί να χειρίστει μεγάλες ποσότητες δεδομένων:** Η Cassandra μπορεί να αποθηκεύσει και να ανακτήσει μεγάλες ποσότητες δεδομένων γρήγορα και αποτελεσματικά, έχει εξαιρετική απόδοση σε ερωτήματα ευρετηρίου και αναζήτησης. Αυτό είναι σημαντικό για εφαρμογές υγείας, οι οποίες συχνά πρέπει να επεξεργάζονται μεγάλες ποσότητες δεδομένων.

### 4.3.2 Σχεδιασμός βάσης δεδομένων

Η Apache Cassandra ως μια NoSQL βάση δεδομένων, επιτρέπει την προσαρμογή των πινάκων ανάλογα με τις συγκεκριμένες ανάγκες των εφαρμογών και των σεναρίων χρήσης. Δυο από τα πιο κρίσιμα στοιχεία που λήφθηκαν υπόψιν κατά τον σχεδιασμό των πινάκων (tables) της βάσης δεδομένων, είναι η επιλογή των κλειδιών και η μη υποστήριξη συνδέσεων (joins) μεταξύ των πινάκων. Αυτό σημαίνει ότι κάθε ερώτημα πρέπει να είναι εξειδικευμένο για έναν μόνο πίνακα, κάτι που καθιστά ακόμη πιο σημαντική την επιλογή των κατάλληλων κλειδιών για τη βελτιστοποίηση της απόδοσης των ερωτημάτων.

### Πίνακας patient\_temp\_registration

patient_temp_registration			
temp_id	pk,PK	uuid	*
email	pk,CK,↑	text	*
first_name		text	
hashed_password		text	
last_name		text	
timestamp		ts	

**Εικόνα 4.2:** Πίνακας patient\_temp\_registration

**Σενάριο Χρήσης:** Αυτός ο πίνακας χρησιμοποιείται για την προσωρινή εγγραφή των ασθενών κατά τη διάρκεια της διαδικασίας εγγραφής.

### **Κλειδιά:**

- Partition Key: temp\_id (UUID)
- Clustering Key: email (TEXT)

### Πίνακας: patients

patients			
patient_uid	pk,PK	text	*
address		text	*
blood_type		text	*
contact_phone		text	*
date_of_birth		dt	*
email		text	*
emergency_contact		text	*
first_name		text	*
hashed_password		text	*
last_name		text	*
refresh_token		text	*

**Εικόνα 4.3:** Πίνακας patients

**Σενάριο Χρήσης:** Ο πίνακας "patients" χρησιμοποιείται για την αποθήκευση των βασικών πληροφοριών των ασθενών.

### **Κλειδιά:**

- Partition Key: patient\_uid (TEXT)

### Πίνακας: doctor\_temp\_registration

doctor_temp_registration			
temp_id	pk,PK	uuid	*
email	pk,CK,↑	text	*
first_name		text	
hashed_password		text	
last_name		text	
specialization		text	
timestamp		ts	

**Εικόνα 4.4:** Πίνακας doctor\_temp\_registration

**Σενάριο Χρήσης:** Αυτός ο πίνακας χρησιμοποιείται για την προσωρινή εγγραφή των γιατρών κατά τη διάρκεια της διαδικασίας εγγραφής.

### **Κλειδιά:**

- Partition Key: temp\_id (UUID)
- Clustering Key: email (TEXT)

### Πίνακας: doctors

doctors			
doctor_id	pk,PK	text	*
email	pk,CK,↑	text	*
address		text	*
designation		text	*
first_name		text	*
hashed_password		text	*
last_name		text	*
phone		text	*
refresh_token		text	*
specialization		text	*
profilepicture		text	

**Εικόνα 4.5:** Πίνακας doctors

**Σενάριο Χρήσης:** Ο πίνακας "doctors" χρησιμοποιείται για την αποθήκευση των βασικών πληροφοριών των επαγγελματιών υγείας.

### **Κλειδιά:**

- Partition Key: doctor\_id (TEXT)
- Clustering Key: email (TEXT)

### Πίνακας: doctor\_patient\_relations

doctor_patient_relations			
:: doctor_id	pk,PK	text	*
:: patient_uid	pk,CK,↑	text	*
:: allergies		text	*
:: contact_phone		text	*
:: date_of_birth		dt	*
:: email		text	*
:: emergency_contact		text	*
:: existing_conditions		text	*
:: family_medical_history		text	*
:: first_name		text	*
:: general_notes		text	*
:: last_name		text	*
:: medications		text	*
:: relationship_notes		text	*
:: surgeries		text	*

**Εικόνα 4.6:** Πίνακας doctor\_patient\_relations

**Σενάριο Χρήσης:** Ο πίνακας "doctor\_patient\_relations" είναι ο πίνακας που χρησιμοποιεί ο επαγγελματίας υγείας για να δημιουργήσει την εγγραφή του ασθενούς στο δικό του σύστημα. Σε αυτόν τον πίνακα, ο επαγγελματίας υγείας μπορεί να καταχωρίσει διάφορες πληροφορίες που αφορούν τον ασθενή, όπως αλλεργίες, υπάρχουσες καταστάσεις, ιατρικό ιστορικό της οικογένειας, φαρμακευτική αγωγή, σημειώσεις σχέσης, χειρουργικές επεμβάσεις και γενικές σημειώσεις. Επίσης, περιλαμβάνει επικοινωνιακές πληροφορίες όπως τηλέφωνο επικοινωνίας, ημερομηνία γέννησης, email και επείγοντα επαφές.

#### **Κλειδιά:**

- Partition Key: doctor\_id
- Clustering Key: patient\_uid

### Πίνακας: visits

visits				
:: doctor_id	pk,PK	text	*	
:: patient_uid	pk,PK	text	*	
:: visit_date	pk,CK, ↑	dt	*	
:: visit_id	pk,CK, ↑	text	*	
:: diagnosis		text	*	
:: diseases		list<str>	*	
:: general_notes		text	*	
:: medicines		list<str>	*	
:: prescription		text	*	
:: symptoms		list<str>	*	
:: visible_to_others		bool	*	
:: visit_title		text	*	
:: visit_type		text	*	
:: attachments		list<str>		
:: billing_info		map<str>		
:: medical_tests		map<str>		
:: referral_info		text		
:: treatment_plan		text		

**Εικόνα 4.7:** Πίνακας visits

**Σενάριο Χρήσης:** Ο πίνακας "visits" είναι ο κεντρικός πίνακας για την καταγραφή και την αναζήτηση επισκέψεων από τους επαγγελματίες υγείας. Επιτρέπει στους επαγγελματίες υγείας να δημιουργούν νέες επισκέψεις, να καταγράφουν διάφορες πληροφορίες όπως συμπτώματα, διάγνωση, και σχέδιο θεραπείας. Οι ασθενείς μπορούν επίσης να δουν το ιστορικό των επισκέψεών τους μέσω αυτού του πίνακα.

### **Κλειδιά:**

- Partition Key: (doctor\_id, patient\_uid)
- Clustering Key: visit\_date, visit\_id

### Πίνακας: shared\_visits

shared_visits				
:: patient_uid	pk,PK	text	*	
:: visit_date	pk,CK, ↓	dt	*	
:: visit_id	pk,CK, ↑	text	*	
:: doctor_id		text	*	
:: doctor_name		text	*	
:: doctor_specialization		text	*	
:: is_visible		bool	*	

**Εικόνα 4.8:** Πίνακας shared\_visits

**Σενάριο Χρήσης:** Ο πίνακας "shared\_visits" συμπληρώνεται κάθε φορά που δημιουργείται μια νέα επίσκεψη. Είναι ο πίνακας που ο ασθενής διαβάζει για να ορίσει αν μια επίσκεψη είναι διαμοιραζόμενη ή όχι.

### Κλειδιά:

- Partition Key: patient\_uid (TEXT)
- Clustering Key: visit\_date, visit\_id

Πίνακας: shared\_visits\_by\_visibility

shared_visits_by_visibility			
::: is_visible	pk,PK	bool	*
::: patient_uid	pk,CK, ↑	text	*
::: visit_date	pk,CK, ↑	dt	*
::: visit_id	pk,CK, ↑	text	*
::: doctor_id		text	*
::: doctor_name		text	*
::: doctor_specialization		text	*

**Εικόνα 4.9:** Πίνακας shared\_visits\_by\_visibility

**Σενάριο Χρήσης:** Ο πίνακας "shared\_visits\_by\_visibility" είναι ο πίνακας που διαβάζουν οι επαγγελματίες υγείας για να δουν ποιες επισκέψεις είναι διαμοιραζόμενες από τον ασθενή και αφορούν άλλους επαγγελματίες υγείας. Επιτρέπει στους επαγγελματίες υγείας να έχουν μια πιο ολοκληρωμένη εικόνα της ιατρικής κατάστασης του ασθενούς.

### Κλειδιά:

- Partition Key: is\_visible (BOOLEAN)
- Clustering Key: patient\_uid, visit\_date, visit\_id

### Πίνακας: entities

entities			
visit_id	pk,PK	text	*
entity_id	pk,CK,↑	text	*
corrected_entity		text	
entity_type		text	
original_entity		text	
timestamp		ts	
version		integer	

**Εικόνα 4.10:** Πίνακας entities

**Σενάριο Χρήσης:** Ο πίνακας "entities" συμπληρώνεται κατά τη δημιουργία ενός "visit" και έχει ως σκοπό την αποθήκευση και ανάλυση κλινικού κειμένου. Ο πίνακας είναι ιδιαίτερα χρήσιμος για την ανάλυση δεδομένων και την εξαγωγή σημαντικών πληροφοριών από τα κλινικά κείμενα, όπως οι διαγνώσεις, τα φάρμακα που έχουν χορηγηθεί, και άλλες σημαντικές παρατηρήσεις.

### **Κλειδιά:**

- Partition Key: visit\_id (TEXT)
- Clustering Key: entity\_id

### Πίνακας: patient\_analysis

patient_analysis			
patient_uid	pk,PK	text	*
diseases_map		map<num>	*
emergency_visits		integer	*
follow_ups		integer	*
medications_map		map<num>	*
routine_visits		integer	*
total_visits		integer	*
visit_details		list<map>	*
notes_modifications		integer	

**Εικόνα 4.11:** Πίνακας patient\_analysis

**Σενάριο Χρήσης:** Ο πίνακας "patient\_analysis" είναι σχεδιασμένος για να παρέχει μια συνοπτική ανάλυση του ιστορικού επισκέψεων και ιατρικών στοιχείων για κάθε ασθενή. Ο πίνακας είναι ιδιαίτερα χρήσιμος για την παρακολούθηση και την ανάλυση της ιατρικής κατάστασης του ασθενούς, καθώς και για την εξαγωγή στατιστικών στοιχείων που μπορούν να χρησιμοποιηθούν για τη βελτίωση της παρεχόμενης φροντίδας.

### **Κλειδιά:**

- Primary Key: patient\_uid (TEXT)

### Πίνακας: spark\_jobs

spark_jobs			
user_id	pk,PK	text	*
job_type	pk,CK, ↑	text	*
completion_time		ts	
job_id		uuid	*
output		text	
status		text	*
submission_time		ts	*
error		text	

**Εικόνα 4.12:** Πίνακας spark\_jobs

**Σενάριο Χρήσης:** Ο πίνακας "spark\_jobs" είναι σχεδιασμένος για να εξυπηρετεί τη διαχείριση και την παρακολούθηση των εργασιών ανάλυσης που εκτελούνται στο Apache Spark. Ο πίνακας είναι ιδιαίτερα χρήσιμος για την παρακολούθηση της προόδου των εργασιών, την αντιμετώπιση προβλημάτων και την ανάλυση των αποτελεσμάτων. Επιπλέον, μπορεί να χρησιμοποιηθεί για την αυτοματοποίηση διαδικασιών και την εξαγωγή στατιστικών στοιχείων.

#### **Κλειδιά:**

- Partition Key: user\_id (TEXT)
- Clustering Key: job\_type

## 4.4 Ανάλυση Διεπαφής FastAPI

Το FastAPI είναι μια διεπαφή λογισμικού και η κεντρική ιδέα πίσω από τον σχεδιασμό αυτής της διεπαφής είναι να ανταποκριθεί αποτελεσματικά στα σενάρια χρήσης και τα ερωτήματα (queries) που προέκυψαν από την αρχιτεκτονική της βάσης δεδομένων που έχουμε ήδη περιγράψει. Με άλλα λόγια, η διεπαφή FastAPI έχει σχεδιαστεί με γνώμονα την αποδοτικότητα, την ευελιξία και την ταχύτητα, ώστε να μπορεί να χειριστεί τα ερωτήματα που προκύπτουν από τα σενάρια χρήσης, τις δυνατότητες αλλά και τους περιορισμούς που καθορίζονται από την αρχιτεκτονική της βάσης δεδομένων. Η διεπαφή FastAPI θα επιτελέσει το ρόλο του backend στο σύστημα μας.

Κάποια επιπλέον θέματα που έχουν ληφθεί υπόψη κατά το σχεδιασμό της διεπαφής είναι:

- CORS (Cross-Origin Resource Sharing):** Η διεπαφή χρησιμοποιεί το πρωτόκολλο CORS για να επιτρέπει μόνο τις εξουσιοδοτημένες εφαρμογές που εκτελούνται σε διαφορετικά domains (όπως η επικοινωνία με το frontend). Αυτό βοηθά στην προστασία της διεπαφής από κακόβουλες επιθέσεις μέσα από το διαδίκτυο.
- Authentication / JWT:** Η διεπαφή χρησιμοποιεί το πρωτόκολλο JWT για να παρέχει ταυτοποίηση και έγκριση στους χρήστες. Αυτό βοηθά στην προστασία των δεδομένων της διεπαφής από μη εξουσιοδοτημένους χρήστες.
- Pydantic models:** Η διεπαφή χρησιμοποιεί το πλαίσιο Pydantic για να δημιουργήσει μοντέλα δεδομένων. Αυτό διευκολύνει την επικοινωνία μεταξύ της διεπαφής και της βάσης δεδομένων και βοηθά στη διασφάλιση ότι τα δεδομένα είναι σωστά μορφοποιημένα και δεν περιέχουν κακόβουλο κώδικα.
- Hashed passwords (bcrypt):** Η διεπαφή χρησιμοποιεί το αλγόριθμο bcrypt για να κωδικοποιήσει τους κωδικούς πρόσβασης των χρηστών. Αυτό βοηθά στην προστασία των κωδικών πρόσβασης από μη εξουσιοδοτημένους χρήστες.

### 4.4.1 Ανάλυση σημείων πρόσβασης (endpoints)

Ο στόχος του σχεδιασμού της διεπαφής FastAPI είναι να προσφερθεί μια ποικιλία endpoints που θα επιτρέπουν στους χρήστες να αλληλοεπιδρούν με τα δεδομένα. Κάθε endpoint έχει μια μοναδική URL (Uniform Resource Locator ) και μια συγκεκριμένη λειτουργία. Τα endpoints που έχουν σχεδιαστεί χωρίζονται σε τρεις κατηγορίες ανάλογα με τον χρήστη που εξυπηρετούν και το έργο τους. Επίσης το FastAPI μας παρέχει τη δυνατότητα να δημιουργήσουμε ένα documentation του api μέσω του Swagger UI, το οποίο παρέχει μια αναλυτική εικόνα των endpoints και αυτήν θα παρουσιάσουμε πιο κάτω.

Αναλυτικά είναι:

## 1. Τα endpoints για τον ασθενή

**Patient**

**POST**

- `POST /patient/token` Login Patient For Access Token / Λήψη διακριτικού πρόσβασης ασθενούς
- `POST /patient/refresh-token` Refresh Patient Token / Ανανέωση διακριτικού ασθενούς
- `POST /patient/registration` Start Registration / Έναρξη εγγραφής
- `POST /patient/finalize-registration` Finalize Registration / Ολοκλήρωση εγγραφής

**GET**

- `GET /patient/check-auth` Check Patient Authentication / Έλεγχος ταυτοποίησης ασθενούς
- `GET /patient/profile` Update Patient Profile / Ενημέρωση προφίλ ασθενούς
- `GET /patient/total_visits` Get Patient Total Visits / Λήψη συνολικών επισκέψεων ασθενούς
- `GET /patient/last_visit` Get Patient Last Visit / Λήψη τελευταίας επίσκεψης ασθενούς
- `GET /patient/shared-visits` Get Shared Visits / Λήψη διαμοιραζόμενων επισκέψεων
- `GET /patient/visit-details` Get Visit Details / Λήψη λεπτομερειών επίσκεψης
- `GET /patient/{patient_uid}` Get Patient / Λήψη πληροφοριών ασθενούς

**PUT**

- `PUT /patient/profile` Update Patient Profile / Ενημέρωση προφίλ ασθενούς
- `PUT /patient/update-visibility` Update Shared Visit Visibility / Ενημέρωση ορατότητας διαμοιραζόμενης επίσκεψης

**Εικόνα 4.13:** Τα endpoints για τον ασθενή

Όπως διακρίνουμε και από το documentation του swagger τα endpoints του ασθενή ομαδοποιούνται βάση του τύπου τους [POST] , [GET] , [PUT].

### [POST]

POST /patient/token: Λήψη διακριτικού/token πρόσβασης ασθενούς.

POST /patient/refresh-token: Ανανέωση διακριτικού/token ασθενούς.

POST /patient/registration: Έναρξη εγγραφής.

POST /patient/finalize-registration: Ολοκλήρωση εγγραφής.

### [GET]

GET /patient/check-auth: Έλεγχος ταυτοποίησης ασθενούς.

GET /patient/profile: Λήψη προφίλ ασθενούς.

GET /patient/total-visits: Λήψη συνολικών επισκέψεων ασθενούς.

GET /patient/last-visit: Λήψη τελευταίας επίσκεψης ασθενούς.

GET /patient/shared-visits: Λήψη διαμοιραζόμενων επισκέψεων.

GET /patient/visit-details: Λήψη λεπτομερειών επίσκεψης.

GET /patient/patient\_uid: Λήψη ασθενούς.

### [PUT]

PUT /patient/profile: Ενημέρωση προφίλ ασθενούς.

PUT /patient/update-visibility: Ενημέρωση ορατότητας διαμοιραζόμενης επίσκεψης.

## 2. Τα endpoints για τον επαγγελματία υγείας

<b>POST</b>	
<b>POST</b>	/doctor/registration Start Registration / Έναρξη εγγραφής
<b>POST</b>	/doctor/register Register New Doctor / Εγγραφή νέου γιατρού
<b>POST</b>	/doctor/token Login For Access Token / Σύνδεση για διακριτικό πρόσβασης
<b>POST</b>	/doctor/refresh-token Refresh Token / Ανανέωση διακριτικού
<b>POST</b>	/doctor/logout Logout / Αποσύνδεση
<b>POST</b>	/doctor/relationship/create Create Relationship / Δημιουργία σχέσης
<b>POST</b>	/doctor/{doctor_id}/patient/{patient_uid}/create-visit Create Visit / Δημιουργία επίσκεψης
<b>GET</b>	
<b>GET</b>	/doctor/check-auth Check Authentication / Έλεγχος ταυτοποίησης
<b>GET</b>	/doctor/profile Get Doctor Profile / Λήψη προφίλ γιατρού
<b>GET</b>	/doctor/relationship/exists Check Relationship Exists / Έλεγχος ύπαρξης σχέσης
<b>GET</b>	/doctor/{doctor_id}/patients Get Patients By Doctor / Λήψη ασθενών από γιατρό
<b>GET</b>	/doctor/{doctor_id}/patient/{patient_uid} Get Patient Details By Uid / Λήψη λεπτομερείων ασθενούς με βάση το Uid
<b>GET</b>	/doctor/{doctor_id}/patient/{patient_uid}/visits Get Visits By Doctor And Patient / Λήψη επισκέψεων από γιατρό και ασθενή
<b>GET</b>	/doctor/{doctor_id}/patient/{patient_uid}/visit/{visit_id}/{visit_date} Get Visits By Doctor And Patient / Λήψη επισκέψεων από γιατρό και ασθενή
<b>GET</b>	/doctor/patient/{patient_uid}/shared-visits Get Shared Visits / Λήψη κοινόχρηστων επισκέψεων
<b>GET</b>	/doctor/{doctor_id}/patient/{patient_uid}/analysis Get Patient Analysis / Λήψη ανάλυσης ασθενούς
<b>PUT</b>	
<b>PUT</b>	/doctor/relationship/update Update Relationship / Ενημέρωση σχέσης

**Εικόνα 4.14:** Τα endpoints για τον επαγγελματία υγείας

Όπως διακρίνουμε και στο documentation του Swagger τα endpoints του επαγγελματία υγείας είναι ομαδοποιημένα βάση του τύπου τους [POST], [GET], [PUT].

### [POST]

POST /doctor/registration: Έναρξη εγγραφής.

POST /doctor/register: Εγγραφή νέου γιατρού.

POST /doctor/token: Σύνδεση για διακριτικό πρόσβασης.

POST /doctor/refresh-token: Ανανέωση διακριτικού.

POST /doctor/logout: Αποσύνδεση.

POST /doctor/relationship/create: Δημιουργία σχέσης.

POST /doctor/doctor\_id/patient/patient\_uid/create-visit: Δημιουργία επίσκεψης.

### [GET]

GET /doctor/check-auth: Έλεγχος ταυτοποίησης.

GET /doctor/profile: Λήψη προφίλ γιατρού.

GET /doctor/relationship/exists: Έλεγχος ύπαρξης σχέσης.

GET /doctor/doctor\_id/patients: Λήψη ασθενών από γιατρό.

GET /doctor/doctor\_id/patient/patient\_uid: Λήψη λεπτομερειών ασθενούς με βάση το

Uid.

GET /doctor/doctor\_id/patient/patient\_uid/visits: Λήψη επισκέψεων από γιατρό και ασθενή.

GET /doctor/doctor\_id/patient/patient\_uid/visit/visit\_id/visit\_date: Λήψη επισκέψεων από γιατρό και ασθενή.

GET /doctor/patient/patient\_uid/shared-visits: Λήψη κοινόχρηστων επισκέψεων. GET /doctor/doctor\_id/patient/patient\_uid/analysis: Λήψη ανάλυσης ασθενούς.

### [PUT]

PUT /doctor/relationship/update: Ενημέρωση σχέσης.

**POST /doctor/register** Register New Doctor / Εγγραφή νέου γιατρού

Register New Doctor / Εγγραφή νέου γιατρού

**Parameters**

No parameters

**Request body** required

application/json

Example Value | Schema

```
{
  "first_name": "string",
  "last_name": "string",
  "email": "string",
  "specialization": "string",
  "address": "string",
  "designation": "string",
  "phone": "string",
  "profilepicture": "string",
  "password": "string"
}
```

**Responses**

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

Media type application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Media type application/json

Controls Accept header.

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

**Εικόνα 4.15:** Request Body in Swagger

## 4.4.2 Ανάλυση χαρακτηριστικών των σημείων πρόσβασης

### Παράμετροι στο URL

Κάποια από τα endpoint που περιγράφαμε πιο πάνω παρουσιάζουν παραμέτρους μέσα στο URL. Για παράδειγμα, η αίτηση GET /doctor/doctor\_id/patient/patient\_uid έχει δύο παραμέτρους: doctor\_id και patient\_uid. Η παράμετρος doctor\_id καθορίζει τον id του γιατρού και η παράμετρος patient\_uid καθορίζει το id του ασθενούς.

Ωστόσο στα endpoints υπάρχει ακόμη ένας τρόπος να στείλουμε δεδομένα και αυτός είναι μέσα από το σώμα (body) όπως για παράδειγμα στην εικόνα για την αίτηση POST /doctor/registration , η οποία έχει παραμέτρους μέσα στο σώμα

```
{
    "first_name": "string",
    "last_name": "string",
    "email": "string",
    "specialization": "string",
    "address": "string",
    "designation": "string",
    "phone": "string",
    "profilepicture": "string",
    "password": "string"
}
```

Μια βασική διαφορά μεταξύ των παραμέτρων που περνάνε μέσα από το URL και των παραμέτρων που περνάνε μέσα από το body είναι ότι οι παράμετροι που περνάνε μέσα από το URL είναι πάντα υποχρεωτικές, ενώ οι παράμετροι που περνάνε μέσα από το body μπορούν να είναι και προαιρετικές.

Επίσης οι παράμετροι URL χρησιμοποιούνται συνήθως για την παροχή πληροφοριών που δεν αλλάζουν συχνά. Για παράδειγμα, το ID ενός γιατρού ή ενός ασθενούς είναι μια σταθερή τιμή που μπορεί να οριστεί ως παράμετρος URL. Ενώ αντίθετα οι παράμετροι του body χρησιμοποιούνται συνήθως για την παροχή πληροφοριών που αλλάζουν συχνά. Για παράδειγμα, τα δεδομένα ενός ασθενούς ή τα δεδομένα μιας επίσκεψης που είναι δυναμικές πληροφορίες, αυτά μπορούν να οριστούν ως παράμετροι του body.

```
▲ A-vou
@app.get("/patient/total_visits", tags=["Patient", "GET"],
          summary="Get Patient Total Visits / Λήψη συνολικών επισκέψεων ασθενούς",
          description="Get Patient Total Visits / Λήψη συνολικών επισκέψεων ασθενούς (Protected endpoint)")
async def get_patient_total_visits(current_patient: dict = Depends(get_current_patient)) -> dict:
    patient_uid = current_patient['patient_uid']

    query = "SELECT COUNT(*) FROM shared_visits WHERE patient_uid = ?"
    prepared = session.prepare(query)
    rows = session.execute(prepared, [patient_uid])
    total_visits = rows[0].count if rows else 0

    return {"total_visits": total_visits}
```

**Εικόνα 4.16:** Endpoint /patient/total-visits

### Authentication / JWT

Για την παροχή ταυτοποίησης και έγκρισης των χρηστών, χρησιμοποιείται το πρωτόκολλο JWT (JSON Web Token). Η διαδικασία αυτή επιτυγχάνεται μέσω της δημιουργίας ενός

διακριτικού πρόσβασης (access token), το οποίο είναι ένα υπογεγραμμένο JSON object που περιέχει πληροφορίες σχετικά με τον χρήστη και το ρόλο του. Το διακριτικό πρόσβασης μπορεί στη συνέχεια να χρησιμοποιηθεί από τον χρήστη για την πρόσβαση σε προστατευμένους πόρους.

```
8 usages  ▲ A-vou *
def get_current_patient(authorization: str = Header(...)) -> dict:
    token = authorization.split(" ")[1] if "Bearer" in authorization else None

    if not token:
        raise credentials_exception

    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        patient_uid = payload.get("sub")
        if patient_uid is None:
            raise credentials_exception
    except JWTError:
        raise HTTPException(status_code=401, detail="Not authenticated")

    print(f"Received token: {token}")
    print(f"Patient UID: {patient_uid}")

    user_data = get_patient_by_id(patient_uid)
    if user_data is None:
        raise HTTPException(status_code=401, detail="User not found")

    return user_data
```

**Εικόνα 4.17:** get\_current\_patient() code

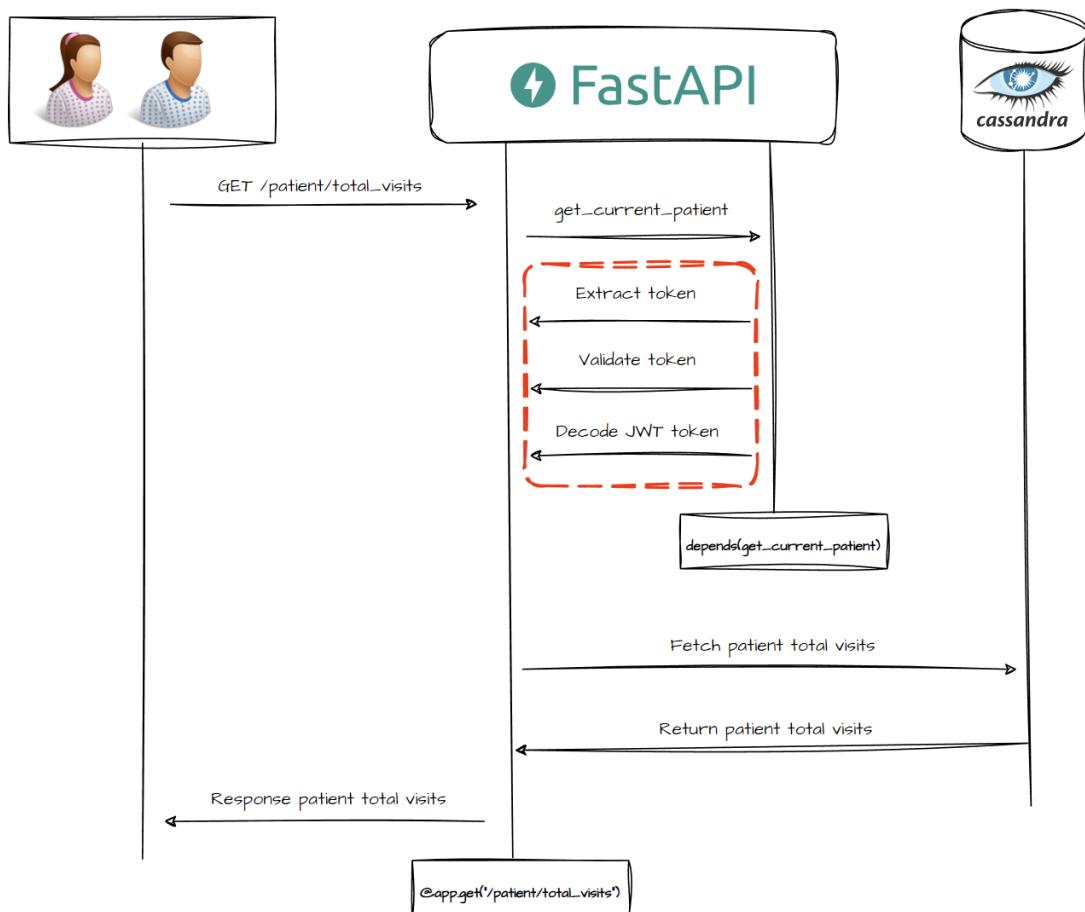
Η διαδικασία ταυτοποίησης και έγκρισης με JWT έχει ως εξής:

- Ο χρήστης στέλνει τα διαπιστευτήριά του στο FastAPI.
- To FastAPI επαληθεύει τα διαπιστευτήρια του χρήστη, δημιουργεί ένα διακριτικό πρόσβασης χρησιμοποιώντας ένα secret key και θέτει ημερομηνία λήξης του token μετά από ένα χρονικό διάστημα.
- To FastAPI επιστρέφει το διακριτικό πρόσβασης στον χρήστη.
- Ο χρήστης στέλνει το διακριτικό πρόσβασης στο FastAPI κάθε φορά που θέλει πρόσβαση σε έναν προστατευμένο endpoint. Συγκεκριμένα το ενσωματώνει στο header του αιτήματος και είναι της μορφής ‘Authorization: Bearer The\_Token’

- Το FastAPI επαληθεύει το διακριτικό πρόσβασης και εάν είναι έγκυρο, επιτρέπει στον χρήστη να πρόσβαση στον πόρο.

Για παράδειγμα το endpoint GET /patient/total-visits: Λήψη συνολικών επισκέψεων ασθενούς. Το συγκεκριμένο endpoint είναι προστατευμένο, καθώς χρησιμοποιεί τη συνάρτηση Depends(get\_current\_patient) για να λάβει τα στοιχεία του τρέχοντα ασθενούς. Η συνάρτηση get\_current\_patient() επαληθεύει το διακριτικό πρόσβασης του ασθενούς και επιστρέφει τα στοιχεία του ασθενούς, εάν το διακριτικό πρόσβασης είναι έγκυρο.

Στο ακόλουθο διάγραμμα παρουσιάζεται ολοκληρωμένη η διαδικασία:



**Εικόνα 4.18:** User-Fastapi-Cassandra diagram

### Pydantic model

Ακολούθως, παρουσιάζεται ένα παράδειγμα υλοποίησης ενός endpoint με Pydantic model

- POST /doctor/relationship/create: Δημιουργία σχέσης επαγγελματία υγείας – ασθενούς.

```

# A-vou *
@app.post("/doctor/relationship/create", response_model=Relationship, tags=["Doctor", "POST"],
          summary="Create Relationship / Δημιουργία σχέσης",
          description="Create Relationship / Δημιουργία σχέσης (Protected endpoint)")
async def create_relationship(data: RelationshipCreate, user: dict = Depends(get_current_user)):
    doctor_id = str(user["doctor_id"])

    try:
        query = """
        INSERT INTO doctor_patient_relations
        (doctor_id, patient_uid, general_notes, allergies, existing_conditions, family_medical_history,
         medications, relationship_notes, surgeries, contact_phone, date_of_birth, email,
         emergency_contact, first_name, last_name)
        VALUES (%s, %s, %s);
        """
        session.execute(query, (
            doctor_id, data.patient_uid, data.general_notes, data.allergies, data.existing_conditions,
            data.family_medical_history, data.medications, data.relationship_notes, data.surgeries, data.contact_phone,
            data.date_of_birth, data.email, data.emergency_contact, data.first_name, data.last_name
        ))

        response_data = {"doctor_id": doctor_id}
        response_data.update(data.dict())
        return response_data
    
```

Εικόνα 4.19: Endpoint /doctor/relationship/create

```

1 usage  # A-vou
class RelationshipCreate(BaseModel):
    patient_uid: str
    allergies: Optional[str] = None
    existing_conditions: Optional[str] = None
    family_medical_history: Optional[str] = None
    medications: Optional[str] = None
    relationship_notes: Optional[str] = None
    surgeries: Optional[str] = None
    general_notes: Optional[str] = None
    contact_phone: Optional[str] = None
    date_of_birth: Optional[date] = None
    email: Optional[str] = None
    emergency_contact: Optional[str] = None
    first_name: str
    last_name: str

```

Εικόνα 4.20: pydantic model

To endpoint αυτό έχει ως σκοπό τη δημιουργία μιας νέας σχέσης μεταξύ ενός επαγγελματία υγείας (γιατρού) και ενός ασθενούς. Μέσω αυτού του endpoint, ο επαγγελματίας υγείας μπορεί να καταχωρίσει διάφορες πληροφορίες για τον ασθενή, όπως γενικές σημειώσεις, αλλεργίες, υπάρχουσες καταστάσεις, ιστορικό οικογενειακών παθήσεων, φαρμακευτική αγωγή,

γή, σημειώσεις σχέσης, χειρουργικές επεμβάσεις και άλλα. Το Pydantic model που ενσωματώνεται στο endpoint εξασφαλίζει την ακρίβεια και την ασφάλεια των δεδομένων. Επιπλέον, διευκολύνει τη διαχείριση των δεδομένων, καθώς καθορίζει τους τύπους δεδομένων και τα πεδία που είναι απαραίτητα (required) ή προαιρετικά (optional).

Το Pydantic model, εκτός από την εξασφάλιση της ασφάλειας, παρέχει μια καθαρή και οργανωμένη δομή των δεδομένων. Αυτό είναι ιδιαίτερα χρήσιμο για την αποτελεσματική διαχείριση και ερώτηση των δεδομένων στη βάση δεδομένων, όπως στην περίπτωση της εισαγωγής της σχέσης του επαγγελματία υγείας με τον ασθενή στον πίνακα doctors\_patient\_relations.

Τέλος με την χρήση του Pydantic model η περιγραφή στο Documentation του swagger γίνεται πιο ολοκληρωμένη και παρέχει περισσότερες πληροφορίες για τον σκοπό και τη χρήση του endpoint.

The screenshot shows the Swagger UI interface for a POST request to the endpoint `/doctor/relationship/create`. The title is "Create Relationship / Δημιουργία σχέσης".

- Parameters:**
  - authorization** (required, string, header): authorization
- Request body required:** application/json
- Example Value | Schema:**

```
{
  "patient_uid": "string",
  "allergies": "string",
  "existing_conditions": "string",
  "family_medical_history": "string",
  "medications": "string",
  "relationship_notes": "string",
  "surgeries": "string",
  "general_notes": "string",
  "contact_phone": "string",
  "date_of_birth": "2023-09-29",
  "email": "string",
  "emergency_contact": "string",
  "first_name": "string",
  "last_name": "string"
}
```

**Εικόνα 4.21:** Pydantic in request body of endpoint in Swagger

The screenshot shows the Pydantic model definition for the `RelationshipCreate` schema. It includes fields such as `patient_uid*`, `allergies`, `existing_conditions`, `family_medical_history`, `medications`, `relationship_notes`, `surgeries`, `general_notes`, `contact_phone`, `date_of_birth`, `email`, `emergency_contact`, `first_name*`, and `last_name*`.

**Εικόνα 4.22:** Pydantic model in Swagger

## 4.5 Ανάπτυξη Frontend με ReactJS

Μέχρι στιγμής, έχουν οριστεί τα σενάρια χρήσης της εφαρμογής, έχει δημιουργηθεί η αρχιτεκτονική της βάσης δεδομένων και έχει αναπτυχθεί η διεπαφή FastAPI (backend). Το τελευταίο στάδιο για την ολοκλήρωση της εφαρμογής, αποτελεί η δημιουργία του γραφικού περιβάλλοντος χρήστη / διεπαφής χρήστη με τη χρήση του ReactJS.

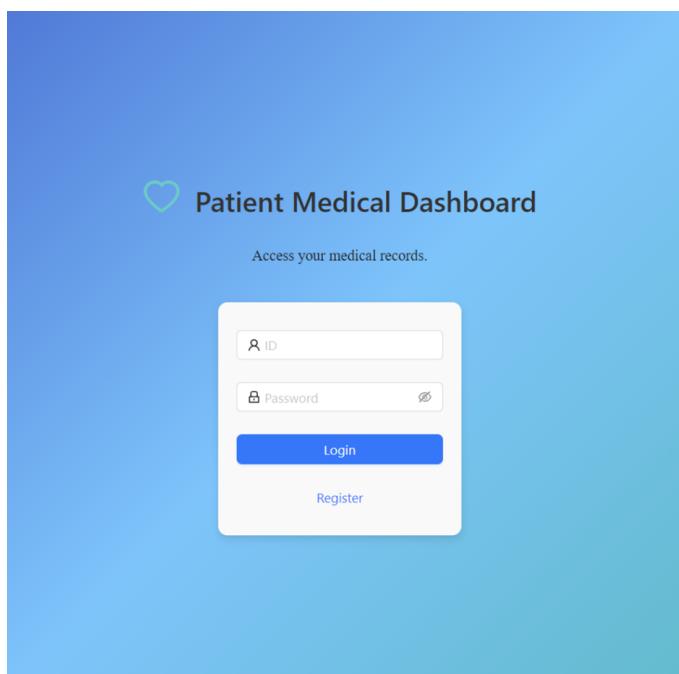
Στο κεφάλαιο αυτό, παρουσιάζεται η ανάπτυξη της διεπαφής χρήστη, μέσω της οποίας επιτρέπεται η εύκολη πρόσβαση και αλληλεπίδραση με τα δεδομένα του συστήματος. Εστιάζοντας, στην υλοποίηση των σεναρίων χρήσης που έχουν ήδη οριστεί, προσφέρεται μια ολοκληρωμένη και φιλική προς τον χρήστη εμπειρία. Με την ενσωμάτωση της ReactJS, δίνεται η δυνατότητα στους χρήστες να έχουν πρόσβαση και να αλληλοεπιδρούν με τα δεδομένα του συστήματος με τρόπο που είναι εύκολος, αποτελεσματικός και ασφαλής.

Έχουν δημιουργηθεί δύο εφαρμογές ReactJS μια για τον ασθενή και μια για τον επαγγελματία υγείας. Κάθε εφαρμογή έχει σχεδιαστεί για να εξυπηρετεί τις συγκεκριμένες ανάγκες του αντίστοιχου χρήστη σύμφωνα και με τα σενάρια χρήσης που έχουμε περιγράψει.

### 4.5.1 Εφαρμογή για τον ασθενή

Η εφαρμογή για τον ασθενή έχει σχεδιαστεί για να παρέχει στον ασθενή πρόσβαση σε πληροφορίες σχετικά με την υγεία του καθώς και να ορίζει ποια δεδομένα θα είναι διαμορφώμενα με τους επαγγελματίες υγείας.

#### Σελίδα εισόδου

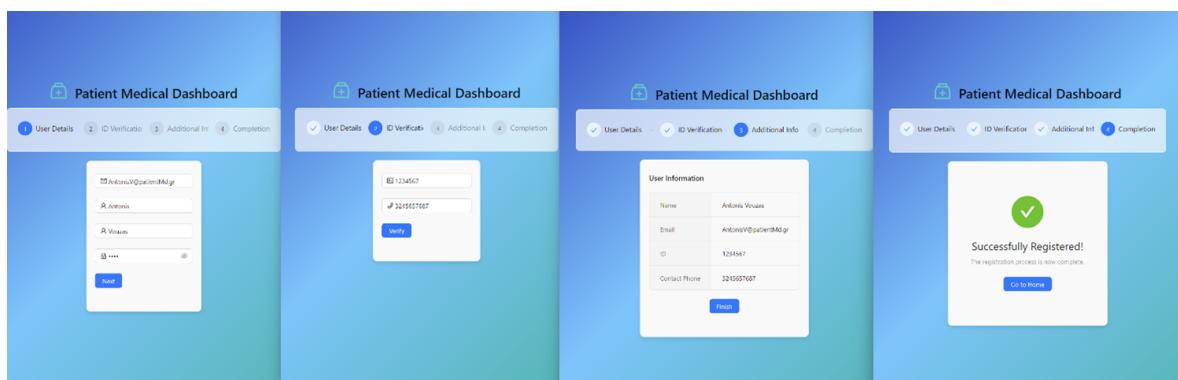


**Εικόνα 4.23:** Είσοδος στην εφαρμογή από τον ασθενή

Η σελίδα εισόδου είναι η πρώτη σελίδα που βλέπει ο χρήστης όταν ανοίξει την εφαρμογή για τον ασθενή. Ο χρήστης μπορεί να συνδεθεί στην εφαρμογή εισάγοντας τον μοναδικό ID (όπως ο AMKA - Αριθμός Μητρώου Κοινωνικής Ασφάλισης) και τον κωδικό πρόσβασής του στα πεδία εισαγωγής. Εάν οι πληροφορίες σύνδεσης είναι σωστές, ο χρήστης θα συνδεθεί στην εφαρμογή και θα μεταφερθεί στην αρχική σελίδα της εφαρμογής.

Αν ο χρήστης δεν έχει λογαριασμό, μπορεί να εγγραφεί κάνοντας κλικ στο κουμπί "Register". Υπάρχουν 3 στάδια για την ολοκλήρωση της εγγραφής του ασθενή

- Στάδιο πρώτο:** εισαγωγή βασικών στοιχείων χρήστη, όνομα, επίθετο, email και κωδικός.
- Στάδιο δεύτερο:** σε αυτό το στάδιο προσομοιώνεται η λειτουργία επιβεβαίωσης στοιχείων από κάποιο πάροχο/οργανισμό μέσω της εισαγωγής του ID και του τηλέφωνου του χρήστη. Αν ο πάροχος ή ο οργανισμός επιβεβαιώσει τον χρήστη, τότε ο χρήστης μπορεί να προχωρήσει στο επόμενο στάδιο.
- Στάδιο τρίτο:** επιβεβαίωση στοιχείων από τον χρήστη
- Στάδιο τέταρτο:** ολοκλήρωση της εγγραφής στο σύστημα.



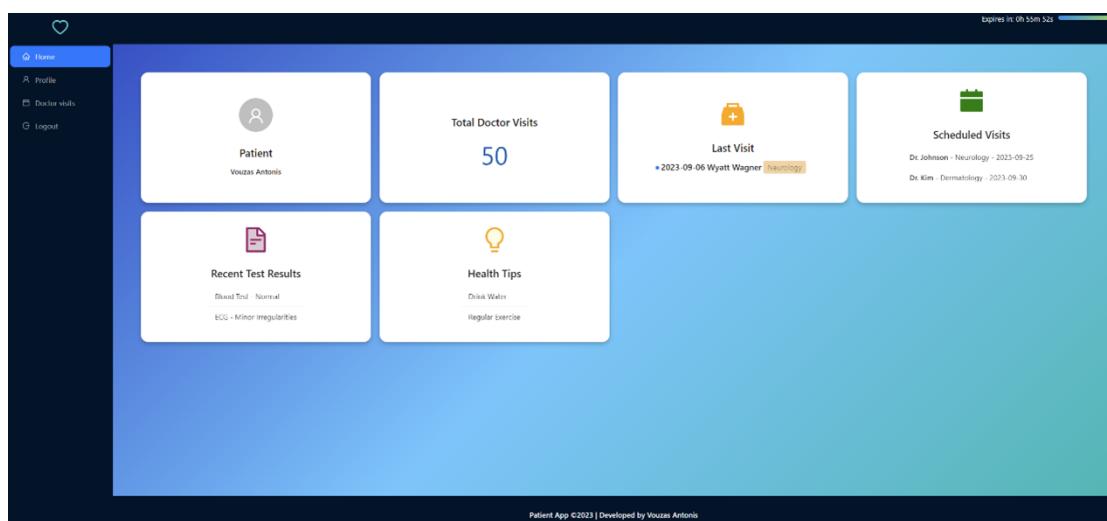
Εικόνα 4.24: Διαδικασία εγγραφής ασθενή

Τα δεδομένα των χρηστών στο πρώτο βήμα αποθηκεύονται προσωρινά στον πίνακα patient\_temp\_registration. Αν ο χρήστης δεν επιβεβαιώσει την εγγραφή του εντός 24 ωρών, τα δεδομένα του διαγράφονται αυτόματα, ενώ στην περίπτωση που τα δεδομένα του επιβεβαιωθούν θα μεταφερθούν στο κύριο πίνακα Patients.

Η διαδικασία εγγραφής με αυτό τον τρόπο προσφέρει ευελιξία καθώς μπορεί να προσαρμοστεί στις παροχές του οργανισμού επικύρωσης. Για παράδειγμα, ο οργανισμός μπορεί να προσφέρει επιπλέον δεδομένα κατά την εγγραφή τα οποία μπορούν να προστεθούν στο πίνακα Patients όπως ασφαλιστικά ταμεία κ.α. Συνολικά, η χρήση δύο πινάκων για την εγγραφή του ασθενή είναι μια αποτελεσματική και ευέλικτη λύση που μπορεί να βοηθήσει στη βελτίωση της ασφάλειας, της ευκολίας χρήσης και της ευελιξίας της διαδικασίας εγγραφής.

## Αρχική σελίδα

Η αρχική σελίδα της εφαρμογής για τον ασθενή παρέχει μια επισκόπηση των βασικών πληροφοριών για τον χρήστη. Ο σκοπός του σχεδιασμού της αρχικής σελίδας είναι να προσφέρει μια γρήγορη εικόνα των προηγούμενων επισκέψεων του ασθενή, των επερχόμενων προγραμματισμένων ραντεβού και να δώσει τη δυνατότητα στον ασθενή να πλοηγηθεί στις σελίδες που επιθυμεί για να δει περισσότερες πληροφορίες.



Εικόνα 4.25: Αρχική σελίδα ασθενή

The screenshot shows the 'Edit Profile' screen of the patient app. The sidebar includes Home, Profile (highlighted), Doctor visits, and Logout. A yellow banner at the top says 'Please Fill Missing Information' with the subtext 'Some of your information is missing. Please fill it in.' Below is a form with fields:

Patient ID	1234567	Date of Birth *	
Address *		Email	AntonisV@patientMd.gr
Blood Type *		Emergency Contact *	
Contact Phone *		First Name	Antonis
		Last Name	Vouzas

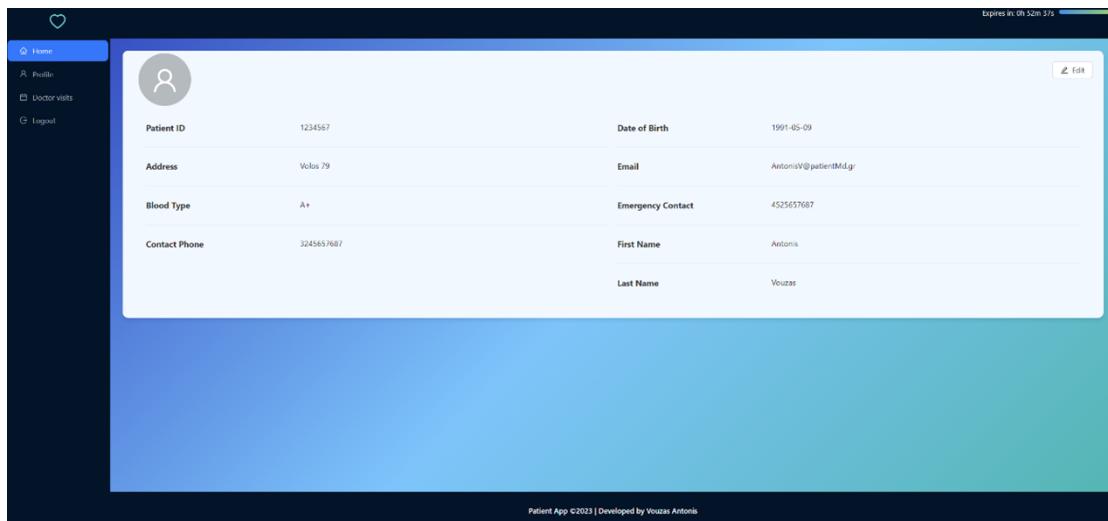
At the bottom center is the text "Patient App ©2023 | Developed by Vouzas Antonis".

Εικόνα 4.26: Ενημέρωση ασθενή για ελλείπει στοιχεία προφίλ

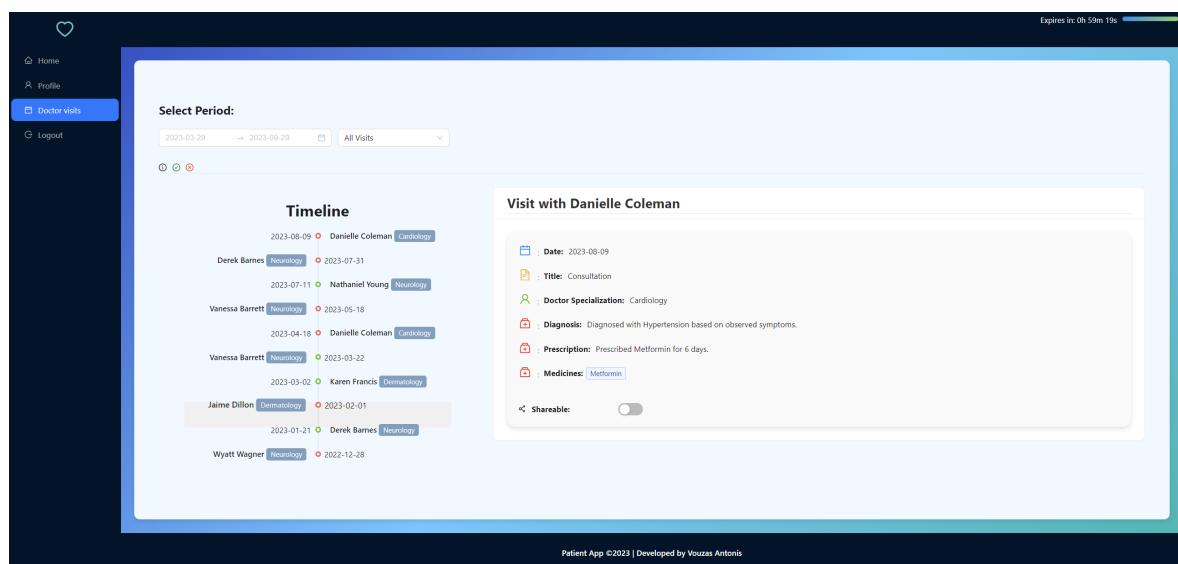
## Σελίδα Προφίλ ασθενούς

Στη σελίδα προφίλ ασθενούς, ο χρήστης μπορεί να δει τα βασικά στοιχεία που εισήγαγε κατά την εγγραφή του. Η σελίδα περιλαμβάνει επίσης πληροφορίες που ο χρήστης μπορεί να εισαγάγει για να συμπληρώσει το προφίλ του. Το σύστημα ενημερώνει τον χρήστη για

την εισαγωγή αυτών των πεδίων ώστε να βελτιώσει την εμπειρία χρήσης του συστήματος. Επιπλέον, εάν κάποιος πάροχος ή οργανισμός παρέχει δεδομένα μετά την ταυτοποίηση του ασθενούς κατά την εγγραφή, αυτά θα είναι ορατά σε αυτή τη σελίδα. Για παράδειγμα το ασφαλιστικό ταμείο είναι ένα πεδίο που θα μπορούσε να συμπληρωθεί από τον πάροχο μετά την ταυτοποίηση του ασθενούς.



**Εικόνα 4.27:** Συμπληρωμένο προφίλ ασθενούς



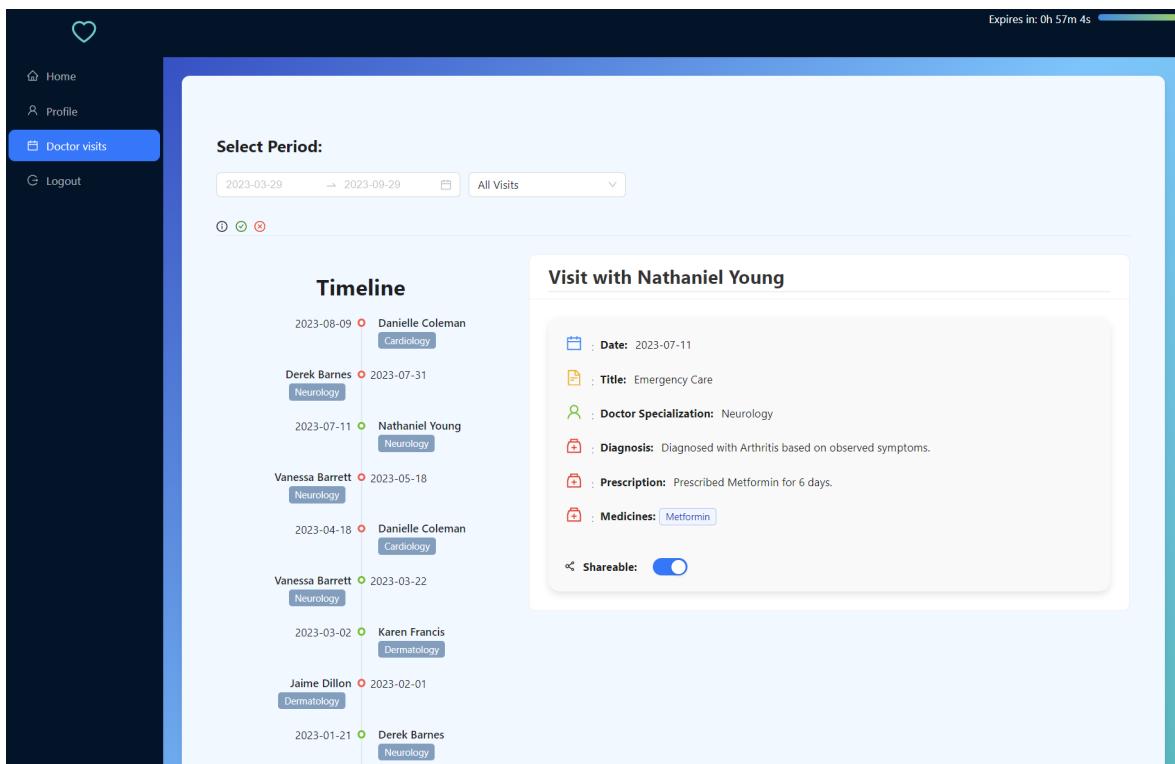
**Εικόνα 4.28:** Μη διαμοιραζόμενη επίσκεψη ασθενούς

## Σελίδα Επισκέψεων σε επαγγελματίες υγείας

Η σελίδα Επισκέψεων σε Επαγγελματίες Υγείας είναι η βασική σελίδα για τον ασθενή, καθώς παρέχει στον ασθενή πλήρη εποπτεία του ιατρικού του ιστορικού. Κάθε επίσκεψη του ασθενή περιλαμβάνει όλες τις σχετικές πληροφορίες, όπως συνταγές, διαγνώσεις, εξετάσεις κ.λπ. Ο ασθενής μπορεί να ανατρέξει σε κάθε επίσκεψη και να δει τις λεπτομέρειες της. Οι

επισκέψεις εμφανίζονται σε ένα χρονοδιάγραμμα (timeline) και η επιλογή μιας επίσκεψης ανοίγει ένα πλαίσιο για την προβολή των λεπτομερειών.

Ένα σημαντικό χαρακτηριστικό της σελίδας αυτής είναι η δυνατότητα του ασθενή να καθορίσει αν μια επίσκεψη θα είναι διαμοιραζόμενη. Αυτό σημαίνει ότι η επίσκεψη θα είναι διαθέσιμη σε όλους τους επαγγελματίες υγείας που έχουν πρόσβαση στο ιστορικό του ασθενή.



**Εικόνα 4.29:** Διαμοιραζόμενη επίσκεψη ασθενούς

#### 4.5.2 Εφαρμογή για τον επαγγελματία υγείας

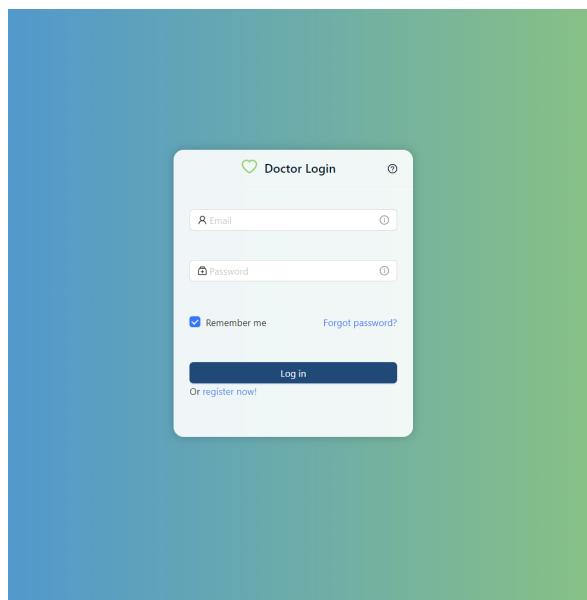
Η εφαρμογή για τον επαγγελματία υγείας έχει σχεδιαστεί για να βελτιώσει την αποτελεσματικότητα και την ποιότητα της υγειονομικής περίθαλψης και για να διευκολύνει τη διαχείριση της φροντίδας των ασθενών. Παρέχει μια ολοκληρωμένη πλατφόρμα στους επαγγελματίες για τη διαχείριση της φροντίδας των ασθενών, συμπεριλαμβανομένων της διαχείρισης ασθενών, της καταγραφής του ιστορικού ασθενών, των προσωποποιημένων αναλύσεων ασθενών και της συνταγογράφησης.

#### Σελίδα Εισόδου

Η σελίδα εισόδου επαγγελματία υγείας είναι η πρώτη σελίδα που εμφανίζεται όταν ένας επαγγελματίας υγείας ανοίξει την εφαρμογή. Η σελίδα παρέχει δύο επιλογές:

- Σύνδεση:** Ο επαγγελματίας υγείας μπορεί να εισάγει το email και τον κωδικό πρόσβασής του για να συνδεθεί στην εφαρμογή.

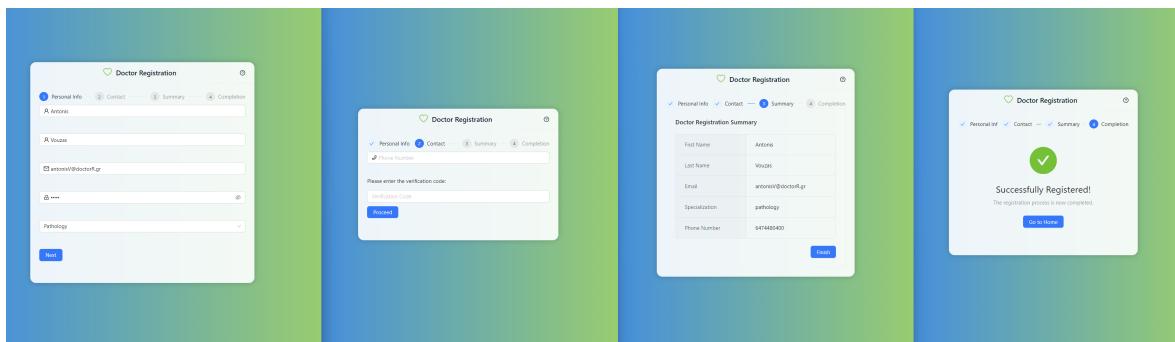
- **Εγγραφή:** Ένας επαγγελματίας υγείας που δεν είναι εγγεγραμμένος μπορεί να επιλέξει αυτήν την επιλογή για να ξεκινήσει τη διαδικασία εγγραφής.



**Εικόνα 4.30:** Είσοδος επαγγελματία υγεία στην εφαρμογή

Εάν ένας επαγγελματίας υγείας δεν είναι εγγεγραμμένος, μπορεί να επιλέξει την επιλογή ”Εγγραφή” (Register) για να ξεκινήσει τη διαδικασία εγγραφής. Η διαδικασία εγγραφής αποτελείται από τέσσερα βήματα:

1. **Εισαγωγή βασικών πληροφοριών:** Ο επαγγελματίας υγείας πρέπει να εισάγει βασικές πληροφορίες, όπως το όνομα, τη διεύθυνση ηλεκτρονικού ταχυδρομείου, τον κωδικό πρόσβασης και την ειδικότητά του.
2. **Επιβεβαίωση επαγγελματία υγείας:** Ο επαγγελματίας υγείας πρέπει να εισάγει τον αριθμό τηλεφώνου του και να λάβει έναν κωδικό επιβεβαίωσης από ένα σύστημα τρίτου μέρους όπως για παράδειγμα κάποιο μητρώο επαγγελματιών υγείας. Η διαδικασία αυτή προσδομοιώνεται στην παρούσα φάση.
3. **Επισκόπηση στοιχείων:** Ο επαγγελματίας υγείας μπορεί να ελέγξει τα στοιχεία που εισήγαγε και να τα διορθώσει εάν είναι απαραίτητο.
4. **Ολοκλήρωση εγγραφής:** Ο επαγγελματίας υγείας πρέπει να αποδεχτεί τους όρους και τις προϋποθέσεις της υπηρεσίας και να πατήσει το κουμπί ”Εγγραφή”.

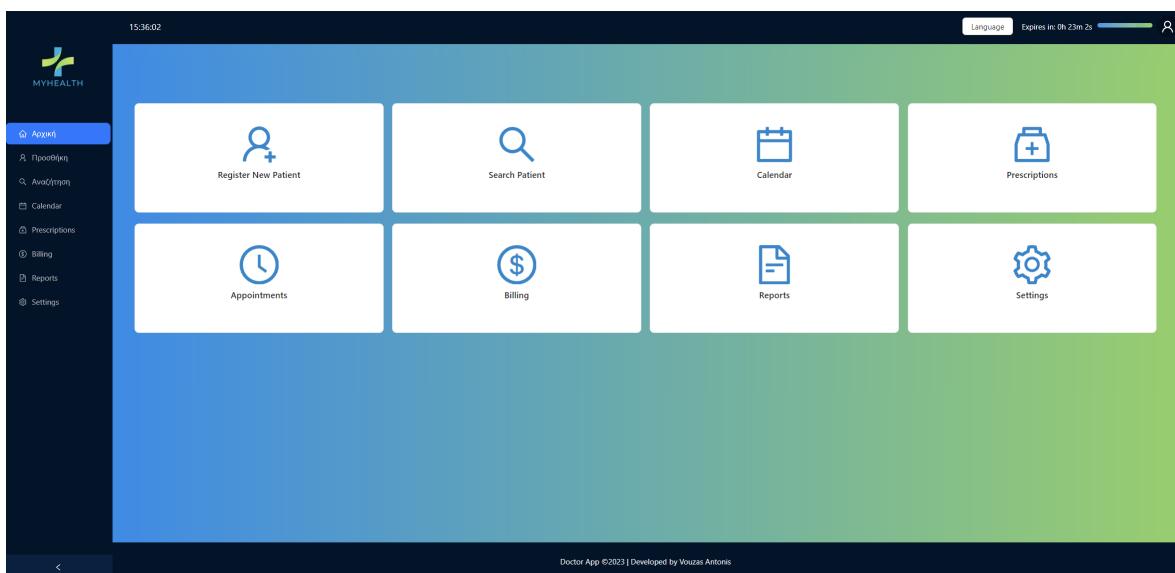


**Εικόνα 4.31:** Βήματα εγγραφής επαγγελματία υγείας

Όπως και στην διαδικασία εγγραφής του ασθενή έχουμε χρήση δυο πινάκων των doctors και doctor temp registration.

## Βήματα εγγραφής επαγγελματία υγείας

Η αρχική σελίδα για τον επαγγελματία υγείας είναι η βασική σελίδα της εφαρμογής και παρέχει πρόσβαση σε όλα τα εργαλεία και τις λειτουργίες που είναι διαθέσιμα.



**Εικόνα 4.32:** Αρχική σελίδα επαγγελματία υγείας

Σελίδα προσθήκης ασθενούς

Η σελίδα προσθήκης ασθενούς είναι μια σελίδα στην εφαρμογή που επιτρέπει στον επαγγελματία υγείας να προσθέσει έναν νέο ασθενή στη λίστα ασθενών του. Η διαδικασία προσθήκης ασθενούς περιλαμβάνει τα εξής βήματα:

- Εισαγωγή ID ασθενούς:** Ο επαγγελματίας υγείας εισάγει το ID του ασθενούς. Το ID του ασθενούς είναι ένα μοναδικό αναγνωριστικό που χρησιμοποιείται για την ταυτοποίηση του ασθενούς στο σύστημα. Η εφαρμογή εμφανίζει τις βασικές πληροφορίες

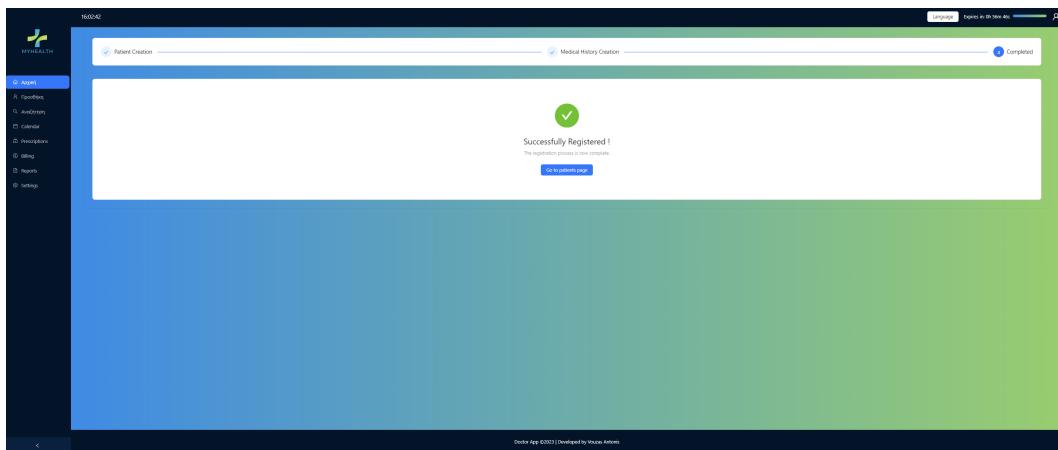
του ασθενούς, όπως το όνομα, το επώνυμο, την ημερομηνία γέννησης και τα υπόλοιπα στοιχεία του ασθενούς που υπάρχουν στο σύστημα.

**Εικόνα 4.33:** Σελίδα προσθήκης ασθενούς

2. **Προσθήκη ιστορικού ασθενούς:** Ο επαγγελματίας υγείας έχει την δυνατότητα να προσθέσει ιστορικό για τον ασθενή, όπως αλλεργίες, εγχειρήσεις, φάρμακα, ασθένειες και ιστορικό οικογένειας.

**Εικόνα 4.34:** Προσθήκη ιστορικών στοιχείων ασθενή

3. **Ολοκλήρωση διαδικασίας:** Ο επαγγελματίας υγείας πατά το κουμπί "Ολοκλήρωση" για να ολοκληρώσει τη διαδικασία προσθήκης ασθενούς.

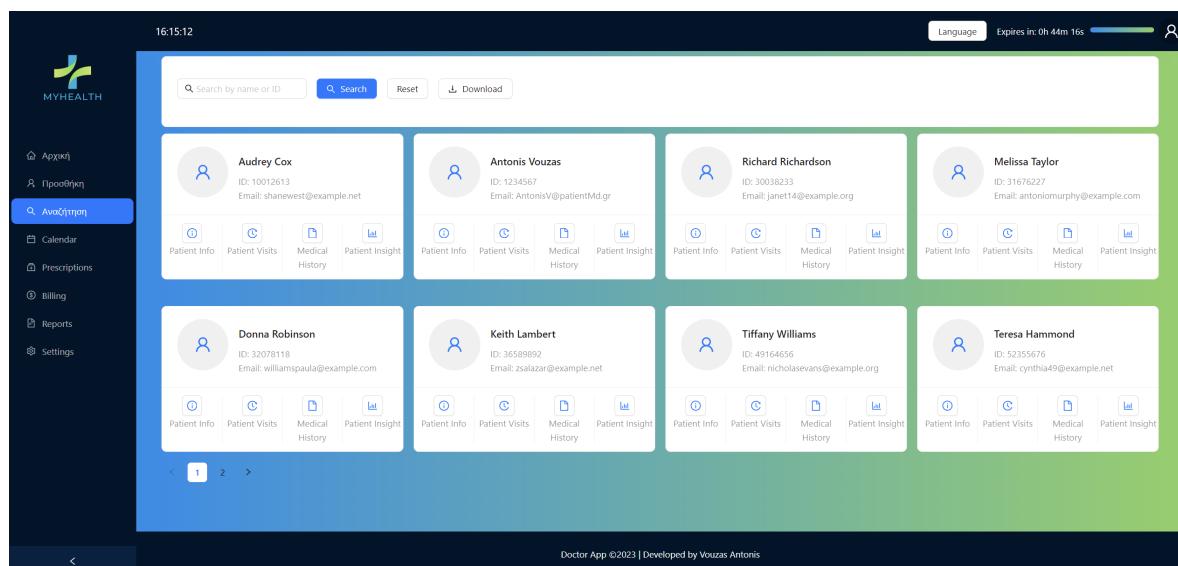


**Εικόνα 4.35:** Ολοκλήρωση εγγραφής νέου ασθενούς

## Σελίδα αναζήτησης ασθενούς

Η σελίδα αναζήτησης ασθενούς είναι μια σελίδα που επιτρέπει στους επαγγελματίες υγείας να αναζητούν ασθενείς στη λίστα ασθενών τους. Η σελίδα περιλαμβάνει μια λίστα με όλους τους ασθενείς του επαγγελματία υγείας, καθώς και μια σειρά φίλτρων που μπορούν να χρησιμοποιηθούν για την αναζήτηση συγκεκριμένων ασθενών.

Ο επαγγελματίας υγείας μπορεί να αναζητήσει έναν ασθενή με βάση το όνομα/επώνυμο του ασθενούς και βάση του μοναδικού αριθμού που χρησιμοποιείται για την ταυτοποίηση του ασθενούς.

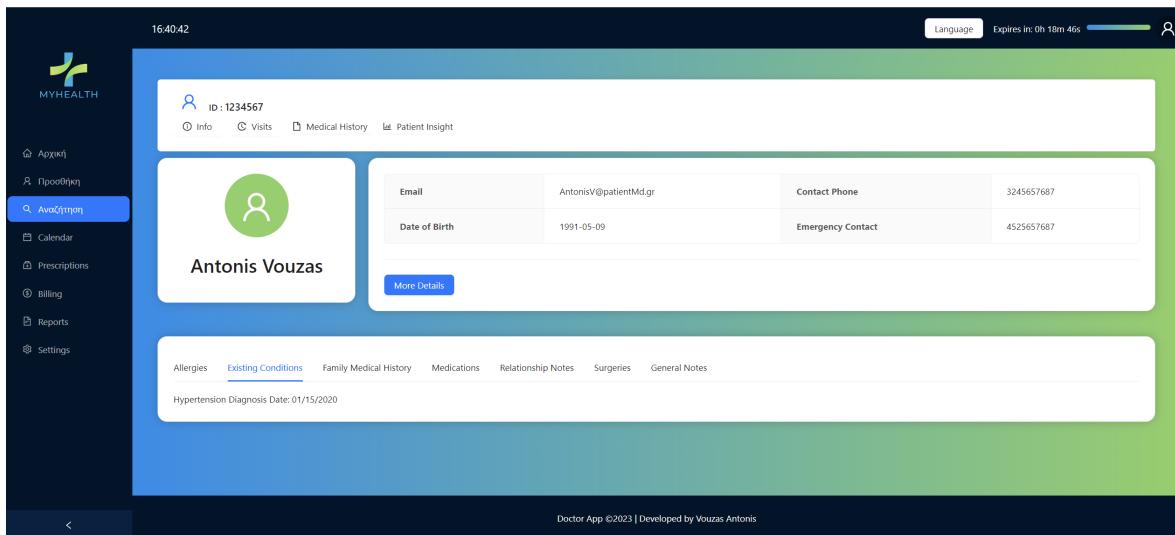


**Εικόνα 4.36:** Αναζήτηση ασθενούς

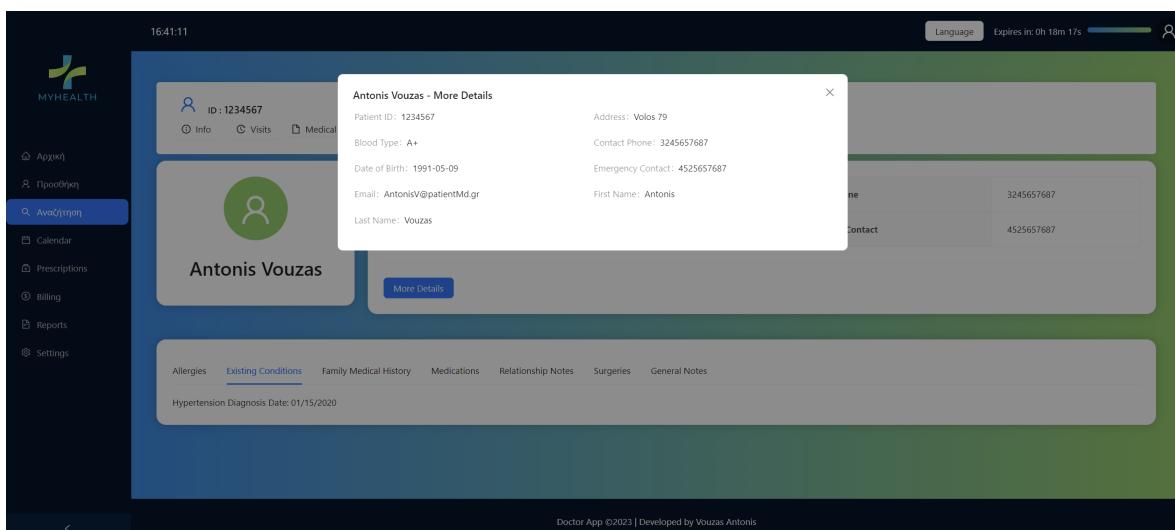
Μόλις ο επαγγελματίας υγείας βρει τον ασθενή που αναζητά, μπορεί να επιλέξει μια από τις παρακάτω επιλογές σχετικά με τον συγκεκριμένο ασθενή.

- 1. Προφίλ ασθενούς:** Οι βασικές πληροφορίες του ασθενούς, όπως το όνομα, το επώνυμο, την ημερομηνία γέννησης, καθώς ένα μικρό ιατρικό ιστορικό ορατό μόνο στον

ίδιο των επαγγελματία υγείας.



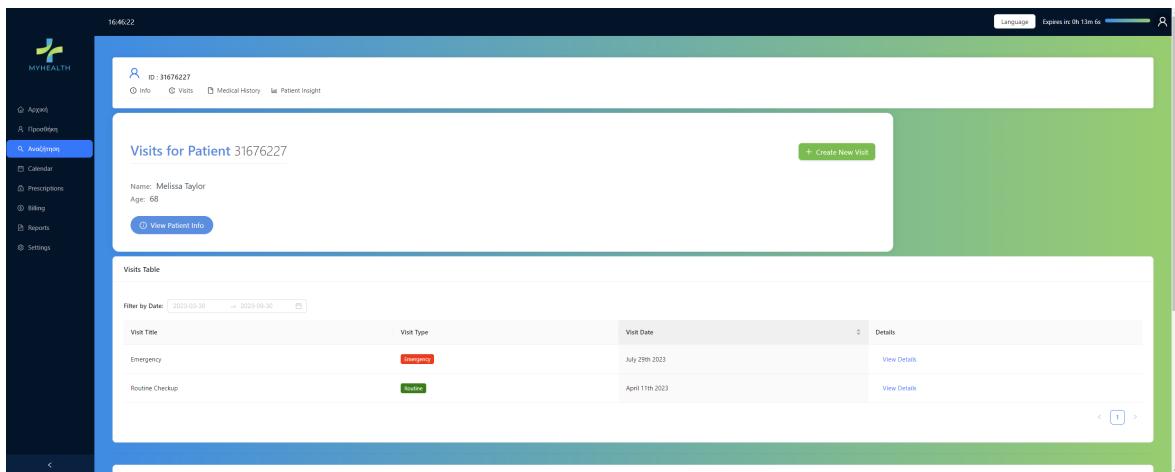
**Εικόνα 4.37:** Προφίλ ασθενούς



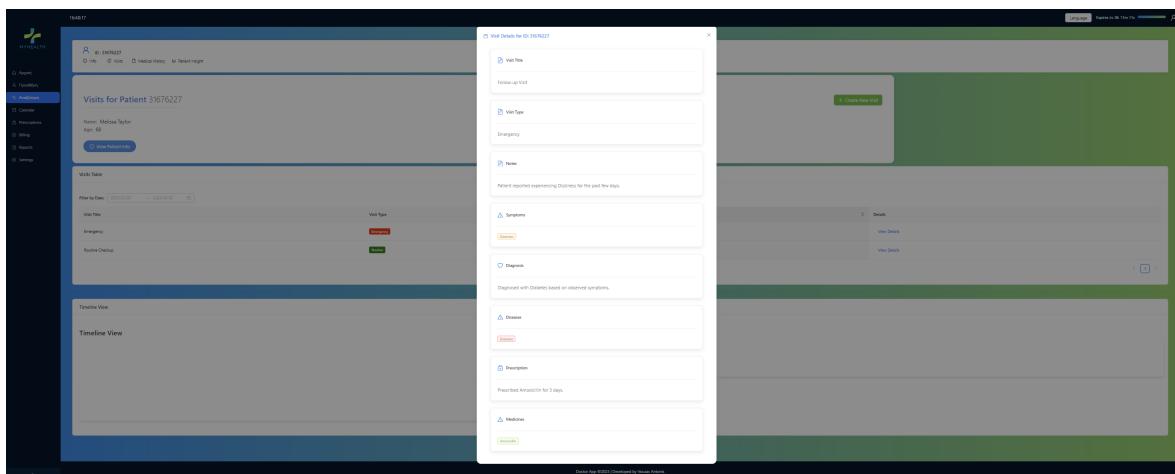
**Εικόνα 4.38:** Επιπλέον στοιχεία προφίλ από Παροχο/Οργανισμό

## 2. Επισκέψεις ασθενούς

- Η σελίδα επισκέψεων ασθενών παρέχει μια λίστα με όλες τις επισκέψεις του ασθενούς, καθώς και ένα χρονοδιάγραμμα για καλύτερη εποπτεία τους. Η λίστα περιλαμβάνει πληροφορίες σχετικά με την ημερομηνία της επίσκεψης, τον τύπο της επίσκεψης και τις σημειώσεις του επαγγελματία υγείας.



Εικόνα 4.39: Επισκέψεις ασθενή



Εικόνα 4.40: Προβολή επίσκεψης ασθενή

- Η σελίδα παρέχει επίσης τη δυνατότητα δημιουργίας νέας επίσκεψης που είναι και η πιο βασική λειτουργία για το σύστημα.

The screenshot shows the MYHEALTH mobile application interface. At the top, there is a dark header bar with the text "16:54:38" on the left and "Language Expires in: 0h 4m 50s" on the right. Below the header is a sidebar on the left containing the following navigation items:

- Αρχική (highlighted in blue)
- Προσέδημα
- Αναζήτηση
- Calendar
- Prescriptions
- Billing
- Reports
- Settings

The main content area is titled "Create Visit for Patient 31676227". It is divided into two sections: "Basic Information" and "Medical Details".

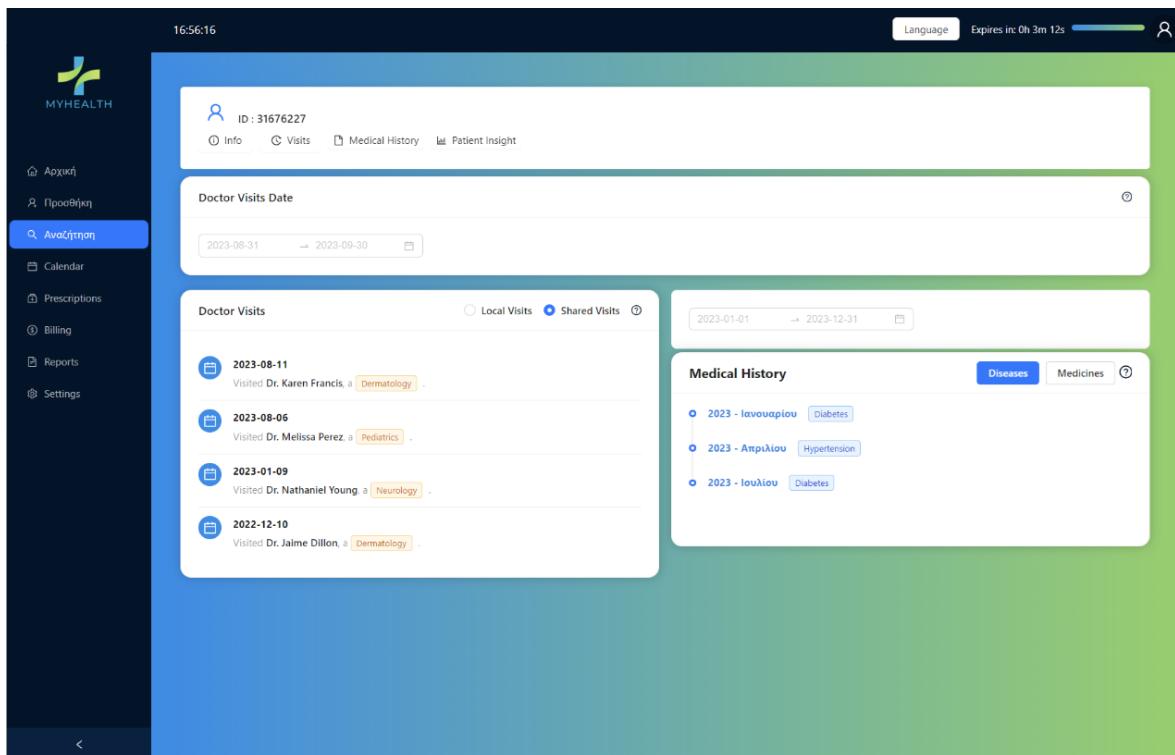
**Basic Information**

- Visit Title: A text input field with placeholder text "e.g., Monthly Checkup".
- Visit Date: A date picker input field labeled "Select Date".
- Visit Type: A dropdown menu labeled "Select visit type".

**Medical Details**

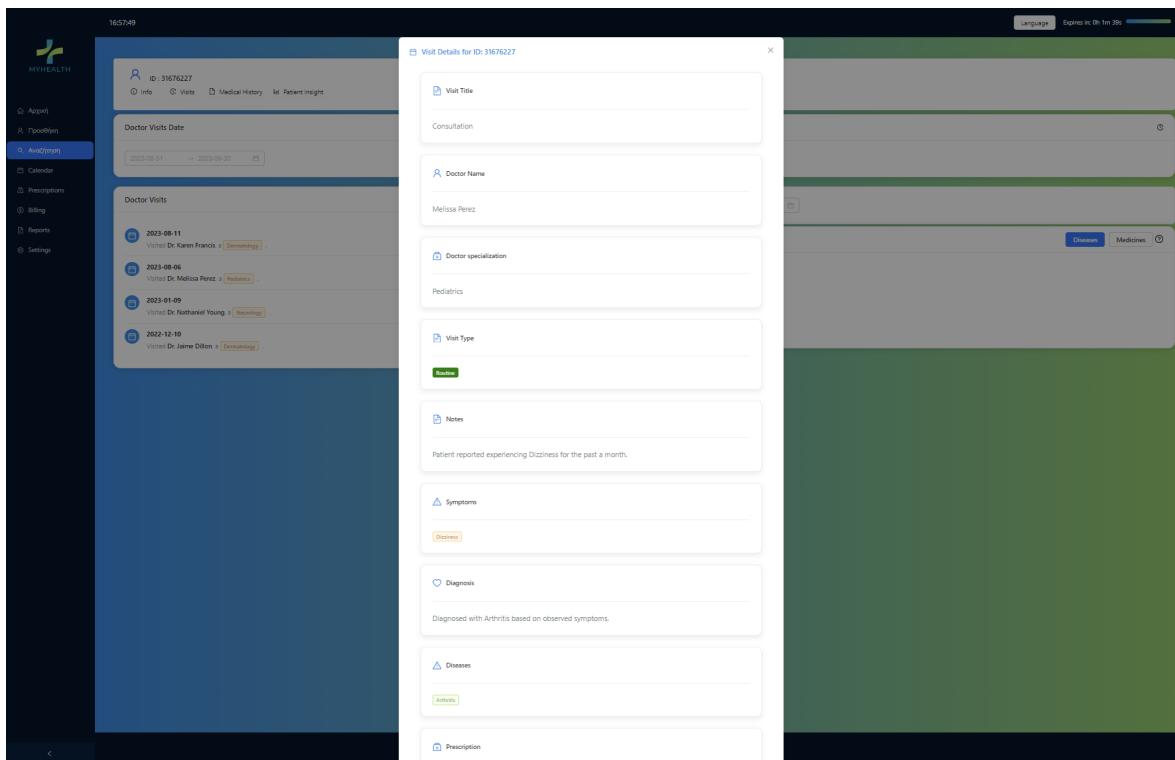
- Diagnosis: A text area labeled "Describe the diagnosis for the visit...".
- Symptoms: A text area labeled "Enter diagnosed symptoms".
- Diseases: A text area labeled "Enter diagnosed diseases".
- Prescription: A text area labeled "Provide prescription details...".
- Medicines: A text area labeled "Medicines".

**Εικόνα 4.41:** Δημιουργία επίσκεψης ασθενή



**Εικόνα 4.42:** Προβολή σελίδας Ιστορικού ασθενή

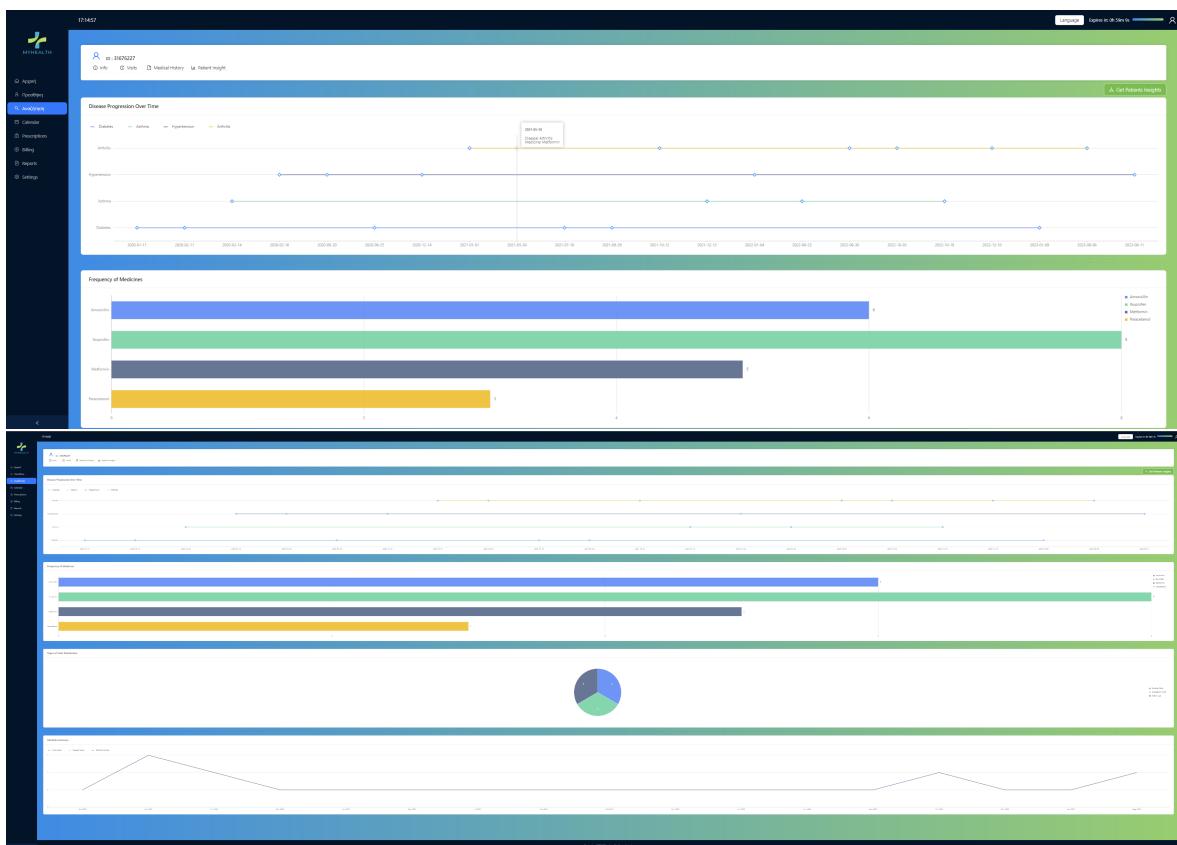
**3. Ιστορικό ασθενούς:** Το ιστορικό του ασθενούς παρέχει μια συγκεντρωτική επισκόπηση όλων των επισκέψεων του ασθενή σε όλους τους επαγγελματίες υγείας. Οι πληροφορίες περιλαμβάνουν την ημερομηνία της επίσκεψης, τον τύπο της επίσκεψης, τις σημειώσεις του επαγγελματία υγείας και τυχόν συνταγογραφήσεις ή διαγνωστικές εξετάσεις που πραγματοποιήθηκαν. Αυτό είναι χρήσιμο για τον επαγγελματία υγείας ώστε να έχει μια ολοκληρωμένη εικόνα της ιατρικής ιστορίας του ασθενούς όπως αυτή καταγράφηκε στο σύστημα. Η σελίδα παρέχει φίλτρα αναζήτησης ώστε να διευκολύνει τον επαγγελματία υγείας.



**Εικόνα 4.43:** Προβολή επίσκεψης σε άλλο επαγγελματία υγείας(διαμοιραζόμενη επίσκεψη)

#### 4. Προσωποποιημένες αναλύσεις/αναφορές ασθενούς

Η σελίδα προσωποποιημένων αναλύσεων/αναφορών παρέχει προσποιημένες αναλύσεις και αναφορές για τον ασθενή όπως συχνότητα ασθενειών, λήψη φαρμάκων, είδος επισκέψεων κ.α. Αυτές οι αναλύσεις και αναφορές μπορούν να χρησιμοποιηθούν για τον εντοπισμό μοτίβων και παραγόντων κινδύνου.



**Εικόνα 4.44:** Διαγράμματα ανάλυσης δεδομένων ασθενή

### 4.5.3 Ανάλυση αρχιτεκτονικής και σχεδιασμού των εφαρμογών

Η αρχιτεκτονική και ο σχεδιασμός των εφαρμογών είναι δύο βασικά στοιχεία που καθορίζουν την επιτυχία τους. Για τον σχεδιασμό της διεπαφής χρήστη (UI) και την καλύτερη εμπειρία του, χρησιμοποιήθηκε η βιβλιοθήκη Ant Design. Η Ant Design είναι μια βιβλιοθήκη που έχει ένα ευρύ φάσμα από συστατικά (components) UI, που καλύπτουν τις περισσότερες απαίτησεις σχεδιασμού καθώς και ένα συνεκτικό και μοντέρνο στυλ που είναι εύκολο στη χρήση και την προσαρμογή του [33].

#### Ant Design Layout

```

return (
  <Layout style={{ minHeight: '100vh' }}>
    <Slider
      breakpoint="lg"
      collapsedWidth="0"
    >
      <div className="logo" style={{ height: '32px', margin: '16px', textAlign: 'center' }}>
        <ImageOutline style={{ fontSize: '32px', color: '#36cfc9' }} />
      </div>
      <Menu theme="dark" mode="inline" defaultSelectedKeys={['home']}>
        <Menu.Item key="home" icon={HomeOutlined}>
          <Link to="/dashboard">Home</Link>
        </Menu.Item>
        <Menu.Item key="1" icon={DiseaseOutlined}>
          <Link to="/profile">Profile</Link>
        </Menu.Item>
        <Menu.Item key="4" icon={CalendarOutlined}>
          <Link to="/visits">Doctor visits</Link>
        </Menu.Item>
        <Menu.Item key="Logout" icon={LogoutOutlined}>
          onClick={handleLogout}
          Logout
        </Menu.Item>
      </Menu>
    </Slider>
    <Layout>
      <Header style={{ padding: 0 }}>
        <div style={{ display: 'flex', justifyContent: 'flex-end' }}>
          <TokenExpirationClock />
        </div>
      </Header>
      <Content style={{ padding: 24, background: COLORS.gradient, minHeight: '360px' }}>
        <div style={{ padding: 0 }}><h1 id="men"></h1></div>
      </Content>
      <Footer style={{ background: "#001529", textAlign: "center" }}><Text style={{ color: "#fffff" }}>Patient App <new Date().getFullYear()> | Developed by <a href="https://www.yous.com">Yous</a></Text></Footer>
    </Layout>
  </Layout>
);
}

3 usages ▾ A-vou
function App() {

```

**Εικόνα 4.45:** Δημιουργία Layout Ant-design

Η βιβλιοθήκη Ant Design παρέχει ένα βασικό layout για εφαρμογές React. Περιλαμβάνει τα ακόλουθα στοιχεία:

- **Header:** Η κεφαλίδα εμφανίζεται στο επάνω μέρος της εφαρμογής. Μπορεί να χρησιμοποιηθεί για να εμφανίσει το λογότυπο της εφαρμογής, το μενού πλοιόγησης και άλλες πληροφορίες.
- **Sider:** Η πλαϊνή μπάρα εμφανίζεται στην αριστερή ή στη δεξιά πλευρά της εφαρμογής. Μπορεί να χρησιμοποιηθεί για να εμφανίσει το μενού πλοιόγησης, τα φίλτρα, τα εργαλεία και άλλες πληροφορίες.
- **Content:** Το περιεχόμενο της εφαρμογής εμφανίζεται στο κεντρικό τμήμα της εφαρμογής.
- **Footer:** Η υποσέλιδα εμφανίζεται στο κάτω μέρος της εφαρμογής. Μπορεί να χρησιμοποιηθεί για να εμφανίσει το πνευματικό δικαίωμα, τους συνδέσμους προς τους όρους χρήσης και την πολιτική απορρήτου της εφαρμογής και άλλες πληροφορίες.

### Επικοινωνία ReactJS με το backend FastAPI

Η επικοινωνία μεταξύ της ReactJS και του backend γίνεται μέσω της διεπαφής του FastAPI. Για να επικοινωνήσει με το backend, η ReactJS χρησιμοποιεί τη βιβλιοθήκη fetch, η οποία παρέχει μια σειρά από μεθόδους που μπορούν να χρησιμοποιηθούν για την υποβολή αιτημάτων HTTP στο backend.

Συγκεκριμένα, όπως έχει ήδη αναφερθεί, τα περισσότερα endpoints του backend είναι προστατευόμενα. Αυτό σημαίνει ότι για να χρησιμοποιηθούν, απαιτείται η αποστολή ενός

token. Το token είναι ένα μοναδικό αναγνωριστικό που επιτρέπει στην εφαρμογή να ταυτοποιηθεί στο backend.

Το token αποστέλλεται στο backend ως μέρος του header του αιτήματος HTTP. Ο ακόλουθος κώδικας δείχνει πώς στέλνετε το token στο backend.

```
useEffect(() :void => {
  usage ± A-vou *
  const fetchPatientData = async () :Promise<void> => {
    const token :string = sessionStorage.getItem('access_token');
    setLoading(true)
    try {
      const response :Response = await fetch(`${apiUrl}/patient/profile`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
          'Accept': 'application/json',
        },
      });
      if (!response.ok) {
        throw new Error('Network response was not ok');
      }
      const data = await response.json();
      setPatient(data);
    } catch (error) {
      notification.error({
        message: 'Error',
        description: 'An error occurred while fetching patient data: ${error.message}',
      });
      console.error('An error occurred while fetching patient data:', error);
    }
    setLoading(false);
  };
  fetchPatientData();
}, []);
```

usage ± A-vou \*

**Εικόνα 4.46:** Παράδειγμα κλήσης endpoint στο Frondent

Σε αυτόν τον javascript κώδικα, η συνάρτηση fetchPatientData() χρησιμοποιεί τη συνάρτηση fetch() για να στείλει μια αίτηση GET στο endpoint /patient/profile του backend. Η συνάρτηση fetch() δέχεται δύο όρους: το URL της αίτησης και ένα αντικείμενο headers που περιέχει τα headers της αίτησης.

Στο αντικείμενο headers, το πεδίο Authorization έχει τιμή Bearer \$token. Η τιμή αυτή δηλώνει ότι η αίτηση είναι εξουσιοδοτημένη με το token που αποθηκεύτηκε στον web browser και συγκεκριμένα στον sessionStorage.

Εάν η αίτηση είναι επιτυχής, η συνάρτηση fetchPatientData() αποθηκεύει τα δεδομένα του ασθενούς στην κατάσταση της εφαρμογής. Εάν η αίτηση αποτύχει, η συνάρτηση εμφανίζει ένα μήνυμα σφάλματος.

Αντός ο κώδικας είναι ένα παράδειγμα του τρόπου με τον οποίο η ReactJS μπορεί να επικοινωνήσει με το backend για να λάβει τα δεδομένα από την βάση δεδομένων.

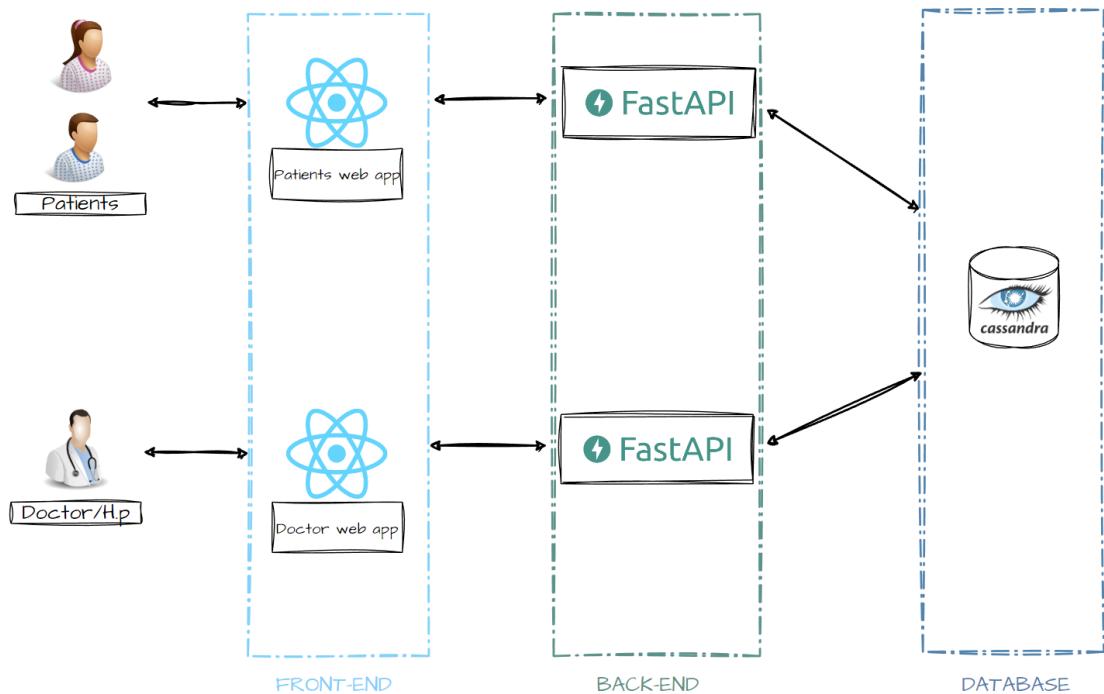
## 4.6 Ενσωμάτωση σε Kubernetes

### 4.6.1 Ανάλυση αρχιτεκτονικής για ενσωμάτωση

Στο κεφάλαιο αυτό, αναλύεται η τελική φάση της εφαρμογής, δηλαδή η ανάπτυξη (deployment) σε ένα σύστημα Kubernetes. Το Kubernetes είναι ένα ιδανικό εργαλείο για να υποστηρίξει τις ανάγκες για κλιμάκωση, διαχείριση και ασφάλεια της εφαρμογής. Στόχος μας, είναι να υποστηρίξουμε εκατομμύρια ασθενείς και χιλιάδες επαγγελματίες υγείας, προσφέροντας ένα υψηλής απόδοσης και αξιόπιστο σύστημα.

Η εφαρμογή είναι το αποτέλεσμα του συνδυασμού τριών κρίσιμων τεχνολογικών στοιχείων:

- **Apache Cassandra:** Η βάση δεδομένων που προσφέρει εξαιρετική ευελιξία και απόδοση, επιτρέποντας την γρήγορη και αποτελεσματική ανάκτηση δεδομένων.
- **FastAPI:** Η διεπαφή (API) που κατασκευάστηκε με την Python, προσφέροντας ταχύτητα, ασφάλεια και ευκολία χρήσης.
- **ReactJS:** Η βιβλιοθήκη JavaScript που χρησιμοποιήσαμε για την δημιουργία ενός δυναμικού και εύχρηστου περιβάλλοντος χρήστη.



Εικόνα 4.47: Αρχιτεκτονική Εφαρμογών

Στον πυρήνα της αρχιτεκτονικής του Kubernetes βρίσκονται τα Pods και τα Nodes, όπως

ήδη έχουμε περιγράψει, τα οποία επιτρέπουν την ευέλικτη και αξιόπιστη λειτουργία των εφαρμογών.

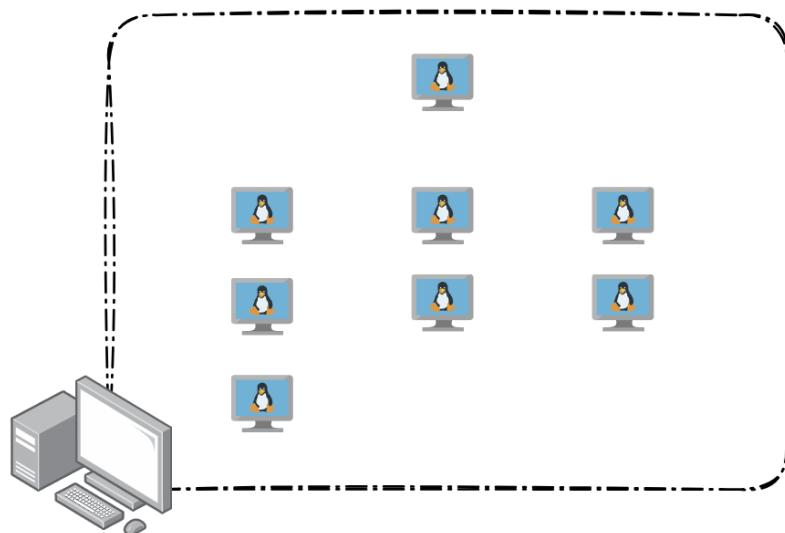
**Κατανεμημένο Σύστημα μέσω Pods και Nodes:** Κάθε Pod είναι ένας αυτόνομος υπολογιστικός κόμβος που μπορεί να φιλοξενήσει ένα ή περισσότερα containers. Τα Pods διανέμονται σε διάφορα Nodes, που είναι οι φυσικοί ή εικονικοί υπολογιστές που συνθέτουν το cluster του Kubernetes. Αυτό επιτρέπει την ομαλή λειτουργία της εφαρμογής, ακόμα και σε περίπτωση αποτυχίας ενός Node, καθώς τα Pods μπορούν να μεταφερθούν αυτόματα σε άλλα Nodes.

**Παράλληλη Λειτουργία μέσω Κλιμάκωσης:** Το Kubernetes προσφέρει αυτόματη κλιμάκωση των Pods με βάση διάφορα κριτήρια, όπως η χρήση CPU και μνήμης. Αυτό σημαίνει ότι μπορούμε να έχουμε πολλαπλές εκδόσεις της ίδιας εφαρμογής να τρέχουν παράλληλα, επιτρέποντας την ταυτόχρονη εξυπηρέτηση εκατομμυρίων αιτημάτων.

Συνοψίζοντας, το Kubernetes επιτρέπει την δημιουργία ενός κατανεμημένου και παράλληλου συστήματος που είναι ικανό να ανταποκριθεί στις αυξανόμενες ανάγκες της εφαρμογής, εξασφαλίζοντας την αποδοτικότητα, την αξιοπιστία και την ευελιξία που απαιτούνται σε ένα σύγχρονο ηλεκτρονικό σύστημα υγείας.

#### 4.6.2 Περιγραφή και δημιουργία του cluster

Για την ανάπτυξη και την ενσωμάτωση της εφαρμογής, χρησιμοποιήθηκε ένα τοπικό περιβάλλον που δημιουργήθηκε στο εργαστήριο. Συγκεκριμένα, δημιουργήθηκαν επτά (7) εικονικές μηχανές (VMs) με το λειτουργικό σύστημα Ubuntu Server 22.0.04.1 LTS [34].



**Εικόνα 4.48:** Εικονικές Μηχανές (VMs)

Οι παράμετροι των εικονικών μηχανών είναι οι εξής:

- Master Node:
  - RAM: 6GB
  - CPUs: 4
  - HDD: 30GB
- Worker Nodes (Kworker1 - Kworker7):
  - RAM: 8GB
  - CPUs: 4
  - HDD: 40GB

Για την επικοινωνία μεταξύ των κόμβων του Kubernetes cluster, δημιουργήθηκε ένα ειδικό VLAN με διεύθυνση IP 192.168.211.0. Το DHCP του VLAN ξεκινάει την ανάθεση διευθύνσεων IP από την 192.168.211.128. Η επικοινωνία με τις εικονικές μηχανές πραγματοποιήθηκε μέσω SSH, εξασφαλίζοντας έναν ασφαλή και αποτελεσματικό τρόπο διαχείρισης.

Για λόγους ευελιξίας και διευκόλυνσης της διαδικασίας εγκατάστασης, δεν εφαρμόστηκαν κανόνες τείχους προστασίας (firewall) στο VLAN. Αντ' αυτού, άνοιξαν όλες οι απαραίτητες πόρτες για την επιτυχή εγκατάσταση και λειτουργία των εφαρμογών του Kubernetes cluster.

#### Τεχνικές Λεπτομέρειες

- VLAN Διεύθυνση: 192.168.211.0
- DHCP Starting IP: 192.168.211.128
- Επικοινωνία: SSH
- Πολιτική Ασφάλειας: Χωρίς ενεργούς κανόνες firewall, ανοιχτές όλες οι απαραίτητες πόρτες

Για την εγκατάσταση και δημιουργία του cluster ακολουθήθηκαν τα βήματα από το επίσημο documentation του Kubernetes [23].

Η διαδικασία δημιουργίας ενός Kubernetes cluster περιλαμβάνει τα εξής βήματα:

- Εγκατάσταση του Kubernetes σε κάθε VM.
- Δημιουργία ενός master node.
- Δημιουργία των workers node.

- Σύνδεση των workers nodes στον master node.

Ο Master Node υπολογιστής χρησιμοποιείται για τη διαχείριση του Kubernetes cluster. Αρχικοποιείται και γίνεται ο διαχειριστής του cluster με την εκτέλεση της εντολής kubeadm init.

Οι Worker Nodes είναι υπεύθυνοι για την εκτέλεση των containers και των εφαρμογών, και για να συνδεθούν στο Master Node, εκτελούν την εντολή kubeadm join, με την οποία γίνεται η εγγραφή τους στο cluster.



**Εικόνα 4.49:** Kubernetes cluster

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
kmaster	Ready	control-plane,master	28d	v1.22.1	192.168.211.220	<none>	Ubuntu 22.04.3 LTS	5.15.0-83-generic	docker://24.0.5
worker1	Ready	<none>	28d	v1.22.1	192.168.211.221	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker2	Ready	<none>	28d	v1.22.1	192.168.211.222	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker3	Ready	<none>	28d	v1.22.1	192.168.211.223	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker4	Ready	<none>	28d	v1.22.1	192.168.211.224	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker5	Ready	<none>	25d	v1.22.1	192.168.211.225	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker6	Ready	<none>	25d	v1.22.1	192.168.211.226	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5
worker7	Ready	<none>	24d	v1.22.1	192.168.211.227	<none>	Ubuntu 22.04.1 LTS	5.15.0-83-generic	docker://24.0.5

**Εικόνα 4.50:** Kubernetes cluster (terminal)

#### 4.6.3 Ενσωμάτωση και Διαμόρφωση της Cassandra

Η εγκατάσταση της Cassandra σε ένα Kubernetes cluster αποτελεί μια σύνθετη διαδικασία που απαιτεί την αυτοματοποίηση και τη διαχείριση διάφορων στοιχείων. Η χρήση

του Helm και του K8ssandra Operator [35] διευκολύνει αυτή τη διαδικασία. Η διαδικασία εγκατάστασης έγινε με τα εξής βήματα:

1. **Εγκατάσταση του Cert-Manager:** Ο Cert-Manager είναι υπεύθυνος για τη διαχείριση των TLS certificates. Είναι ένα απαραίτητο στοιχείο για την ασφάλεια της επικοινωνίας μεταξύ των nodes της Cassandra.
2. **Εγκατάσταση του Operator:** Ο Operator αυτοματοποιεί τη διαχείριση της Cassandra στο Kubernetes. Απλοποιεί τη διαδικασία εγκατάστασης, αναβάθμισης και συντήρησης.
3. **Διαμόρφωση των PVs:** Για να αποθηκεύσουμε τα δεδομένα της Cassandra, δημιουργήσαμε Persistent Volumes (PVs) για κάθε node. Κάθε PV αντιστοιχεί σε ένα node και παρέχει χώρο αποθήκευσης για τα δεδομένα της Cassandra που φιλοξενεί αυτός ο node. Η λογική που ακολουθήσαμε είναι ότι κάθε node του Kubernetes που θα αποτελέσει Node της Cassandra, θα αποτελέσει και φυσικό χώρο αποθήκευσης για τα δεδομένα της Cassandra. Αυτό μας επιτρέπει να έχουμε μια πιο συνεπή και αποδοτική αρχιτεκτονική για το εργαστήριο μας.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-kworker1
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /mnt/local-storage-kworker1
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - worker1
```

Εικόνα 4.51: Enter Caption

4. **Δημιουργία των Namespaces:** Η χρήση των namespaces βοηθά στην οργάνωση και τον περιορισμό των πόρων, καθώς και στην απομόνωση των εφαρμογών. Για την καλύτερη οργάνωση δημιουργήσαμε το namespace my-k8ssandra και k8ssandra-operator.
5. **Δημιουργία του Cassandra Cluster:** Τέλος, δημιουργήσαμε το Cassandra cluster χρησιμοποιώντας ένα YAML αρχείο που περιγράφει τις απαιτούμενες παραμέτρους.

```

apiVersion: k8ssandra.io/v1alpha1
kind: K8ssandraCluster
metadata:
  name: k8ssandra
  namespace: my-k8ssandra
spec:
  auth: false
  cassandra:
    serverVersion: "4.0.1"
    datacenters:
      - metadata:
          name: dc1
          size: 4
        storageConfig:
          cassandraDataVolumeClaimSpec:
            storageClassName: local-storage
            accessModes:
              - ReadWriteOnce
            resources:
              requests:
                storage: 5Gi
      config:
        jvmOptions:
          heapSize: 512M
      stargate:
        size: 1
        heapSize: 256M

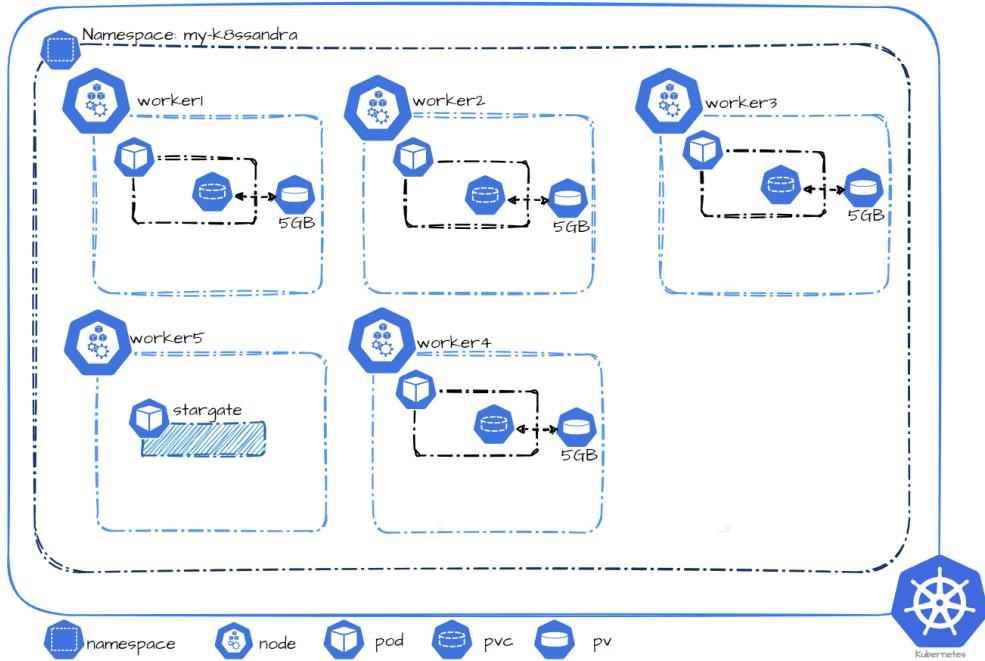
```

Εικόνα 4.52: K8ssandra deployment

Με την εφαρμογή του συγκεκριμένου YAML αρχείου, δημιουργήσαμε ένα Cassandra cluster με το όνομα "dc1". Αυτό το Cassandra cluster αποτελείται από 4 nodes, και κάθε node έχει διαθέσιμο χώρο αποθήκευσης 5GB. Επιπλέον δημιουργήθηκε και ένα pod με την υπηρεσία Stargate.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
k8ssandra-dc1-default-stargate-deployment-78cdd86f75-29vdk	1/1	Running	7 (12d ago)	16d	10.244.42.93	worker5
k8ssandra-dc1-defaultsts-0	2/2	Running	0	12d	10.244.189.84	worker2
k8ssandra-dc1-defaultsts-1	2/2	Running	0	12d	10.244.235.145	worker1
k8ssandra-dc1-defaultsts-2	2/2	Running	0	12d	10.244.199.165	worker4
k8ssandra-dc1-defaultsts-3	2/2	Running	0	12d	10.244.182.23	worker3

Εικόνα 4.53: Pods of K8ssandra in namespace (terminal)



Εικόνα 4.54: Cassandra in Kubernetes

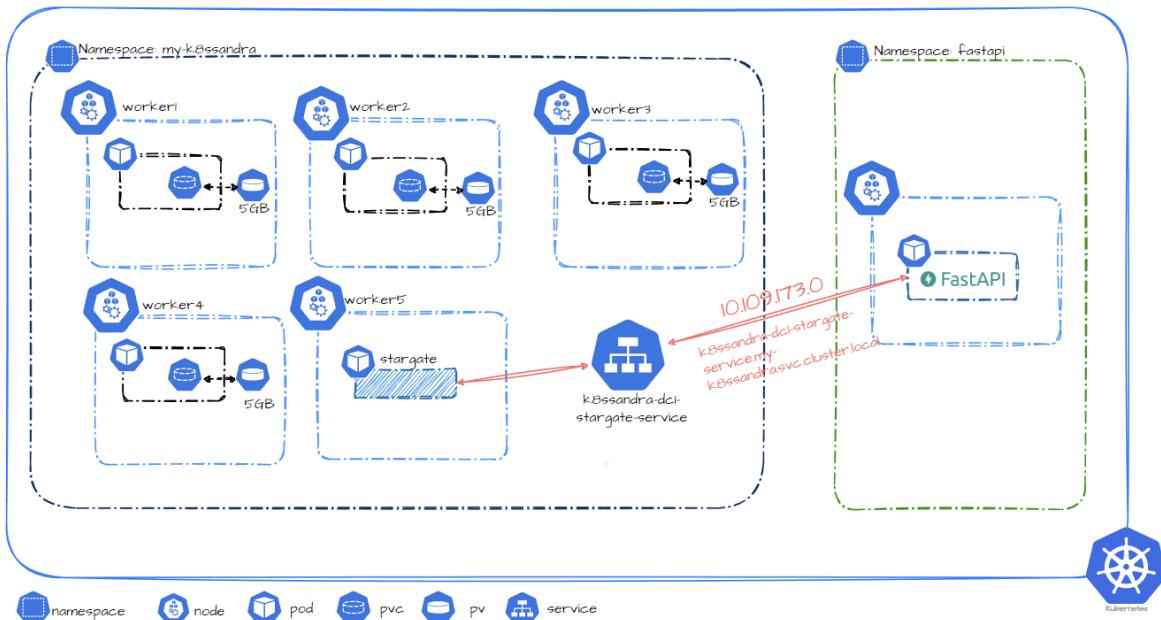
To Stargate είναι μια υπηρεσία που επιτρέπει στους χρήστες να αποκτούν πρόσβαση στη βάση δεδομένων Cassandra μέσω μιας διαδικτυακής διεπαφής χρήστη (UI) - API. Επίσης προσφέρει μια σειρά από άλλα χαρακτηριστικά που διευκολύνουν τη χρήση και τη διαχείριση της βάσης δεδομένων Cassandra [36].

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
k8ssandra-dc1-additional-seed-service	ClusterIP	None	<none>	<none>	16d
k8ssandra-dc1-all-pods-service	ClusterIP	None	<none>	9042/TCP, 8080/TCP, 9103/TCP, 9000/TCP	16d
k8ssandra-dc1-service	ClusterIP	None	<none>	9042/TCP, 9142/TCP, 8080/TCP, 9103/TCP, 9000/TCP	16d
k8ssandra-dc1-stargate-service	ClusterIP	10.109.173.0	<none>	8080/TCP, 8081/TCP, 8082/TCP, 8084/TCP, 8085/TCP, 8090/TCP, 9042/TCP	16d
k8ssandra-seed-service	ClusterIP	None	<none>	<none>	16d

Εικόνα 4.55: Services in namespace my-k8ssandra (terminal)

H Cluster IP είναι η IP διεύθυνση μέσω της οποίας οι εφαρμογές και οι υπηρεσίες εντός του cluster μπορούν να επικοινωνήσουν μεταξύ τους. Στην εγκατάσταση μας (εικόνα 4.55) η Cluster IP για την επικοινωνία με το Stargate και κατ' επέκταση με την βάση δεδομένων Cassandra είναι 10.109.173.0. Ωστόσο εντός του Kubernetes cluster, μπορούμε να χρησιμοποιήσουμε το όνομα της υπηρεσίας ως domain name για την επικοινωνία, αντί για την Cluster IP, διότι είναι προτιμότερο μιας και το όνομα της υπηρεσίας παραμένει σταθερό, ενώ η Cluster IP μπορεί να αλλάξει.

Επομένως, αν η υπηρεσία λέγεται k8ssandra-dc1-stargate-service όπως στην εικόνα και βρίσκεται στο namespace my-k8ssandra, τότε μπορούμε να την καλέσουμε με το FQDN (Fully Qualified Domain Name) k8ssandra-dc1-stargate-service.my-k8ssandra.svc.cluster.local εντός του cluster. Με αυτόν τον τρόπο θα προσεγγίσουμε και την σύνδεση του Backend (Fastapi) με την Cassandra.



**Εικόνα 4.56:** Cassandra service deployment

### Ασφάλεια και Δικτύωση

Για λόγους ευκολίας, δεν έχουν εφαρμοστεί κανόνες firewall και όλες οι απαιτούμενες θύρες είναι ανοιχτές. Αυτό είναι τεχνικά σωστό για ένα περιβάλλον δοκιμών, αλλά σε ένα κανονικό περιβάλλον, θα πρέπει να εφαρμοστούν αυστηρότερες πολιτικές ασφάλειας.

#### 4.6.4 Ενσωματωση και Διαμόρφωση των εφαρμογών (Backend-Frontend)

Για την ολοκλήρωση της εγκατάστασης και της διαμόρφωσης των εφαρμογών backend και frontend, εξασφαλίζοντας την επικοινωνία μεταξύ τους και με τους εξωτερικούς χρήστες, ασθενείς / επαγγελματίες υγείας, ακολουθήθηκαν τα εξής βήματα:

- Εγκατάσταση και Διαμόρφωση του Backend (Doctor Backend)

1. **Dockerization:** Δημιουργήθηκε ένα Docker image για το backend (fastapi).

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim-buster

WORKDIR /app

COPY . /app

# Install Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 8001

# Run app.py when the container launches
CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8001"]
```

**Εικόνα 4.57:** Dockerfile of fastapi

2. **Kubernetes Deployment:** Δημιουργήθηκε ένα YAML αρχείο (doctor-backend-deployment.yaml) για το deployment του backend στο Kubernetes cluster.

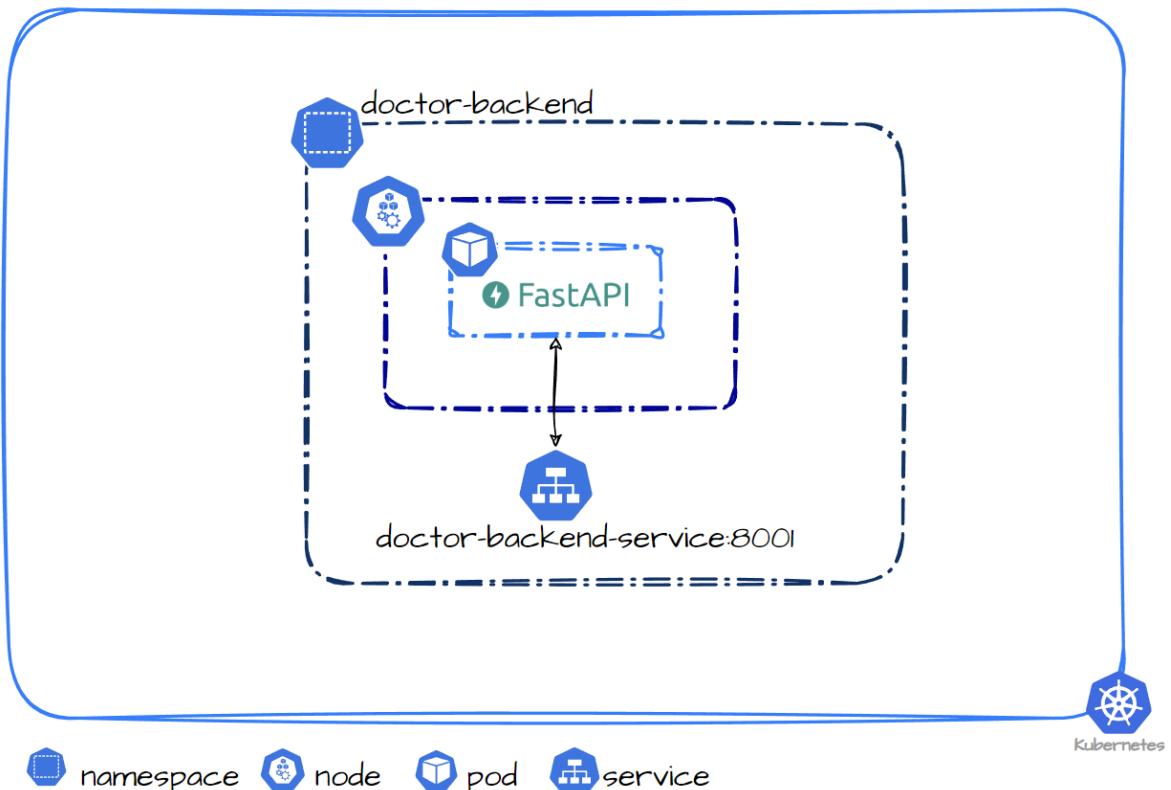
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: doctor-backend
  namespace: doctor-backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: doctor-backend
  template:
    metadata:
      labels:
        app: doctor-backend
    spec:
      containers:
        - name: doctor-backend
          image: antvouz/doctor-backend:v1.3
          ports:
            - containerPort: 8001
      imagePullSecrets:
        - name: my-dockerhub-secret
```

**Εικόνα 4.58:** Fastapi of doctor deployment file

3. **Service:** Δημιουργήθηκε ένα Kubernetes Service (doctor-backend-service.yaml) για να εκτεθεί το backend στο εσωτερικό δίκτυο του cluster.

```
apiVersion: v1
kind: Service
metadata:
  name: doctor-backend-service
  namespace: doctor-backend
spec:
  selector:
    app: doctor-backend
  ports:
    - protocol: TCP
      port: 8001
      targetPort: 8001
      type: ClusterIP
```

**Εικόνα 4.59:** Deployment of service (doctor fastapi)



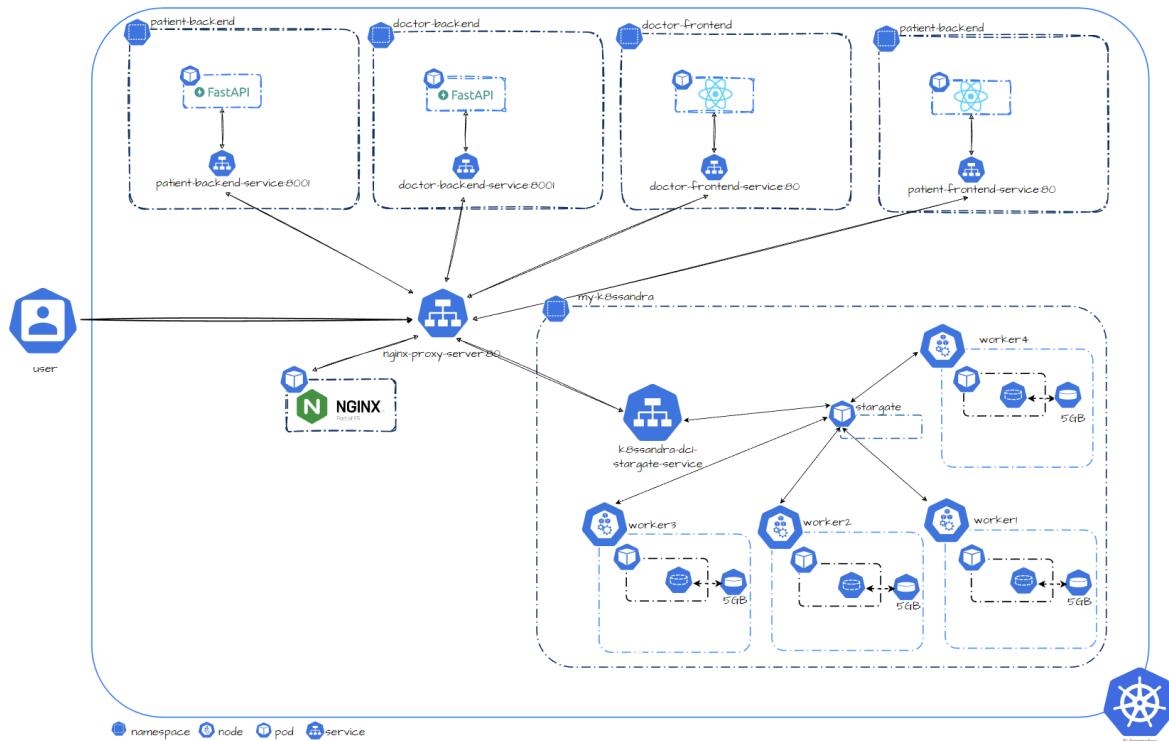
**Εικόνα 4.60:** Fastapi service in Kubernetes

Ομοίως εγκαταστάθηκαν και διαμορφώθηκαν το Doctor frontend, το Patient frontend και backend.

Για να επιτευχθεί η εσωτερική επικοινωνία μεταξύ του backend και του frontend κάθε εφαρμογής αλλά και η επικοινωνία του χρήστη με αυτές ενσωματώθηκε στο σύστημα το Nginx ως reverse proxy.

Ορίστηκαν οι κανόνες στο nginx.config και δημιουργήθηκε το nginx-proxy-server service ώστε να είναι δυνατή η διαχείριση των εισερχόμενων και εξερχόμενων συνδέσεων. Με αυτόν τον τρόπο ο nginx λειτουργεί ως ενιαίο σημείο εισόδου (entry point) για όλες τις εφαρμογές, διασφαλίζοντας την ασφάλεια και την απόδοση του συστήματος.

- **Εσωτερική Επικοινωνία:** Οι εφαρμογές backend και frontend επικοινωνούν μεταξύ τους μέσω του nginx, χρησιμοποιώντας τα ονόματα των Services και που έχουν οριστεί στο nginx.config.
- **Εξωτερική Επικοινωνία:** Οι εξωτερικοί χρήστες συνδέονται στο nginx, το οποίο δρομολογεί τα requests προς τις κατάλληλες εφαρμογές καθώς και τα replies τους με βάση τους κανόνες που έχουν οριστεί.



**Εικόνα 4.61:** Nginx in Kubernetes

Τέλος η επιλογή της συγκεκριμένης αρχιτεκτονικής Kubernetes προσφέρει μια σειρά από οφέλη που είναι καίρια για την αποδοτικότητα, την ευελιξία και την αξιοπιστία του συστήματος.

Η χρήση των YAML αρχείων για τη διαμόρφωση των deployments και των services επιτρέπει την εύκολη και γρήγορη κλιμάκωση των εφαρμογών. Με απλές αλλαγές στα αρχεία αυτά, ο χρήστης μπορεί να αυξήσει τον αριθμό των pods για κάθε εφαρμογή χωρίς να επηρεάσει τη λειτουργία του συστήματος, ώστε να μπορεί έτσι το σύστημα να ανταποκριθεί σε αυξημένες απαιτήσεις φορτίου ή σε αποτυχίες υποδομής.

## 4.7 Ανάλυση με Spark

Με την ολοκλήρωση της ενσωμάτωσης του συστήματος και την ευελιξία που προσφέρει το Kubernetes στον έλεγχο των πόρων, ανοίγει ο δρόμος για την ενσωμάτωση προηγμένων εργαλείων ανάλυσης δεδομένων, όπως ο Apache Spark. Η ενσωμάτωση του Apache Spark με την Cassandra είναι ιδιαίτερα σημαντική, καθώς προσφέρει δυνατότητες που δεν είναι δυνατόν να επιτευχθούν μόνο με τη χρήση της απλής CQL (Cassandra Query Language). Με το Spark, προσφέρετε η δυνατότητα να εκτελεστούν σύνθετες ερωτήσεις και αναλύσεις, όπως συναρτησιακές αναλύσεις, μηχανική μάθηση, και γραφικές αναλύσεις, πάνω στα δεδομένα που αποθηκεύονται στη Cassandra.

### 4.7.1 Ενσωματωση και Διαμόρφωση Spark

Η ενσωμάτωση του Apache Spark στο σύστημα πραγματοποιήθηκε μέσω ενός Helm chart της Bitnami [37], μιας αξιόπιστης πηγής για τη διαχείριση πακέτων στο Kubernetes. Η αρχιτεκτονική του Spark που χρησιμοποιήθηκε είναι η "standalone", προσαρμοσμένη ωστόσο για λειτουργία εντός ενός Kubernetes cluster.

- Master Node: Το Master node λειτουργεί ως ένα ανεξάρτητο pod μέσα στο Kubernetes cluster και είναι υπεύθυνο για τον συντονισμό και την κατανομή των εργασιών στα Worker nodes.
- Worker Nodes: Οι Workers λειτουργούν επίσης ως ανεξάρτητα pods μέσα στο cluster και εκτελούν τις εργασίες που τους ανατίθενται από το Master node.

Αυτή η διάταξη προσφέρει δύο βασικά πλεονεκτήματα:

1. Ευελιξία στη Διαχείριση Πόρων: Η δυνατότητα προσθήκης ή αφαίρεσης Worker nodes επιτρέπει την καλύτερη χρήση των πόρων.
2. Αποδοτική Εκτέλεση Εργασιών: Η αρχιτεκτονική επιτρέπει την παράλληλη εκτέλεση των εργασιών, καθιστώντας την ανάλυση δεδομένων πιο αποδοτική.

Με την ολοκλήρωση της εγκατάστασης δημιουργήθηκαν ένα master pod και δύο worker pods στο namespace my-spark.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
my-spark-master-0	1/1	Running	3 (14d ago)	25d	10.244.35.23	worker6	<none>	<none>
my-spark-worker-0	1/1	Running	3 (14d ago)	25d	10.244.42.92	worker5	<none>	<none>
my-spark-worker-1	1/1	Running	0	14d	10.244.168.194	worker7	<none>	<none>

Εικόνα 4.62: Spark setup

The screenshot shows the Apache Spark UI interface. At the top, it displays the URL: `Spark Master at spark://my-spark-master-0.my-spark-headless.my-spark.svc.cluster.local:7077`. Below this, there's a summary of system resources:

- URL: `spark://my-spark-master-0.my-spark-headless.my-spark.svc.cluster.local:7077`
- Alive Workers: 2
- Cores in use: 8 Total, 0 Used
- Memory in use: 6.3 GiB Total, 0.0 B Used
- Resources In use:
- Applications: 0 Running, 17 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Below the summary, there are two sections:

- Workers (2)**: A table showing two workers with their addresses, states, cores, and memory usage.
- Running Applications (0)**: A table showing zero running applications.

**Εικόνα 4.63:** Spark UI

Η χρήση του Python framework του FastAPI και της Python-βασισμένης βιβλιοθήκης PySpark επιτρέπει τη δημιουργία μιας ολοκληρωμένης αρχιτεκτονικής συστήματος για δυναμική χρήση των δυνατοτήτων του Spark.

Συγκεκριμένα, δημιουργήθηκε ένας δυναμικός μηχανισμός για την εκτέλεση εργασιών ανάλυσης με τον Spark μέσω του FastAPI.

### Αναλυτικός Σχεδιασμός του Pipeline:

#### 1. Δημιουργία PySpark Templates:

- Έχουν δημιουργηθεί Python αρχεία (.py) που περιέχουν όλο τον απαραίτητο κώδικα για την εκτέλεση των αναλύσεων σε Spark (PySpark jobs).
- Αυτά τα αρχεία λειτουργούν ως “templates” και ενσωματώνονται στον κώδικα του FastAPI.

#### 2. Σύνδεση με Endpoints:

- Κάθε PySpark template συνδέεται με ένα συγκεκριμένο FastAPI endpoint.
- Όταν αυτό το endpoint καλείται, ενεργοποιείται μια αλυσίδα διαδικασιών που οδηγούν στην εκτέλεση του αντίστοιχου PySpark template ως ένα PySpark job.

#### 3. Δυναμική Παραμετροποίηση:

- Το FastAPI επιτρέπει την προσθήκη δυναμικών παραμέτρων, όπως το UID του ασθενούς.
- Αυτές οι παράμετροι μεταβιβάζονται δυναμικά στα PySpark templates, επιτρέποντας εξατομικευμένη εκτέλεση των jobs.

#### 4. Αποθήκευση και Προβολή Δεδομένων:

- Αφού ολοκληρωθεί η ανάλυση, τα αποτελέσματα αποθηκεύονται στη βάση δεδομένων Apache Cassandra.

- Τα αποτελέσματα γίνονται διαθέσιμα στον επαγγελματία υγείας μέσω της frontend εφαρμογής, που εχει δημιουργηθεί σε ReactJS.

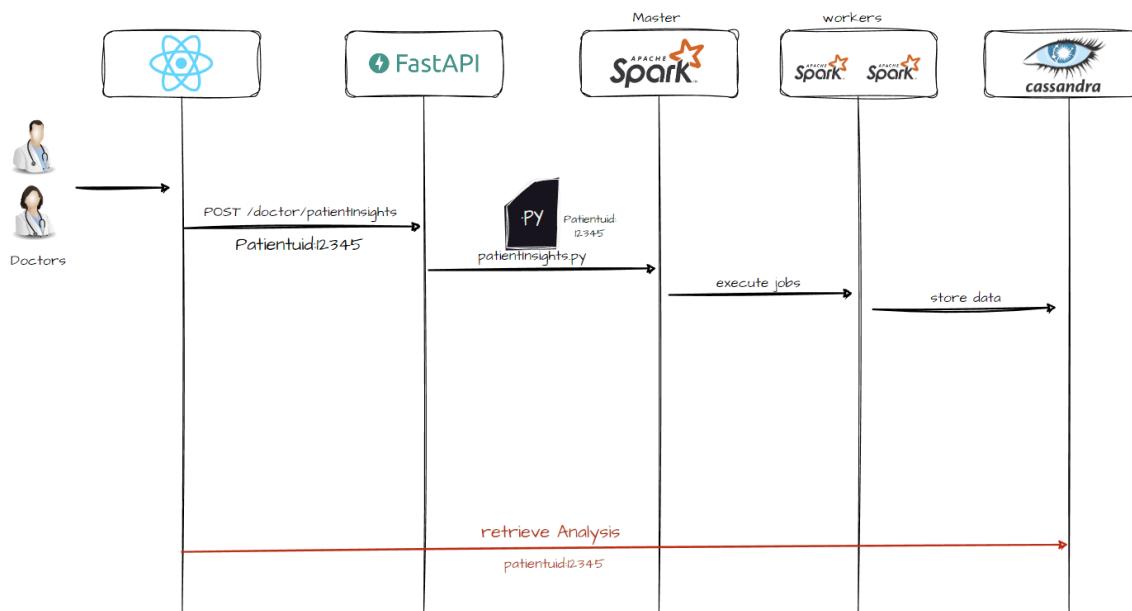
### 5. Διαχείριση Jobs:

- Χρησιμοποιείτε ένας πίνακας spark\_jobs (Εικόνα 4.12) για τον έλεγχο της δημιουργίας και της εκτέλεσης των Spark jobs.
- Αυτό παρέχει ένα επιπλέον επίπεδο διαχείρισης και εποπτείας των εργασιών.

#### 4.7.2 Ανάλυση δεδομένων με Spark

Για παράδειγμα, στην εφαρμογή, ο επαγγελματίας υγείας έχει τη δυνατότητα να εκκινήσει διάφορους τύπους αναλύσεων δεδομένων μέσω του frontend. Κάθε τύπος ανάλυσης αντιστοιχεί σε ένα διαφορετικό endpoint και, κατ' επέκταση, σε ένα διαφορετικό PySpark template. (Εικόνα 4.64).

1. Επιλογή Τύπου Ανάλυσης: Ο επαγγελματίας υγείας επιλέγει τον τύπο ανάλυσης που επιθυμεί (π.χ. "Patient Insights") και πατά το αντίστοιχο κουμπί στο frontend.
2. Αίτηση στο Endpoint: Το frontend εκτελεί μια POST αίτηση στο επιλεγμένο endpoint. Στο σώμα της αίτησης περιλαμβάνεται το UID του ασθενούς (π.χ. UID:12345).
3. Δυναμική Εισαγωγή UID στο Template: Στο backend, το FastAPI λαμβάνει το UID και το εισάγει δυναμικά στο αντίστοιχο PySpark template. Αυτό το template περιέχει ήδη όλο τον απαραίτητο κώδικα για την εκτέλεση της επιλεγμένης ανάλυσης.
4. Υποβολή στο Spark Master: Το πλήρως προετοιμασμένο PySpark αρχείο κώδικα υποβάλλεται στον Spark Master.
5. Εκτέλεση και Ανάλυση: Ο Spark Master και οι workers διαβάζουν τα απαραίτητα δεδομένα από τη βάση δεδομένων Apache Cassandra, εκτελούν την ανάλυση και αποθηκεύουν τα αποτελέσματα πίσω στην Apache Cassandra.
6. Προβολή Αποτελεσμάτων: Τα αποτελέσματα της ανάλυσης γίνονται διαθέσιμα στον επαγγελματία υγείας μέσω του frontend. (Εικόνα 4.66)

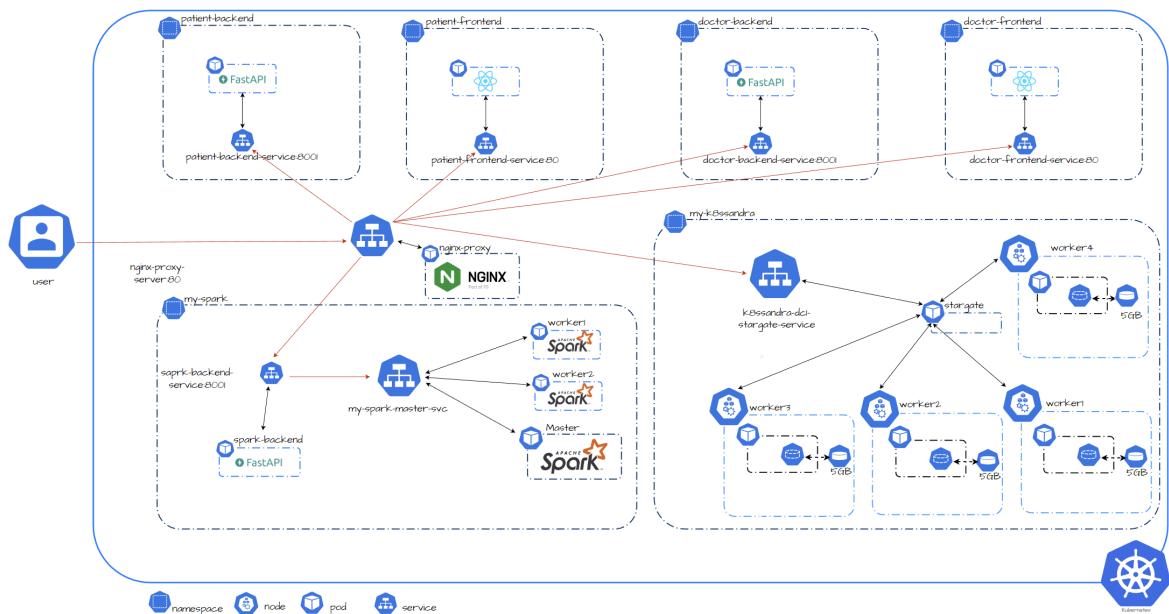


**Εικόνα 4.64:** Spark analysis pipeline example

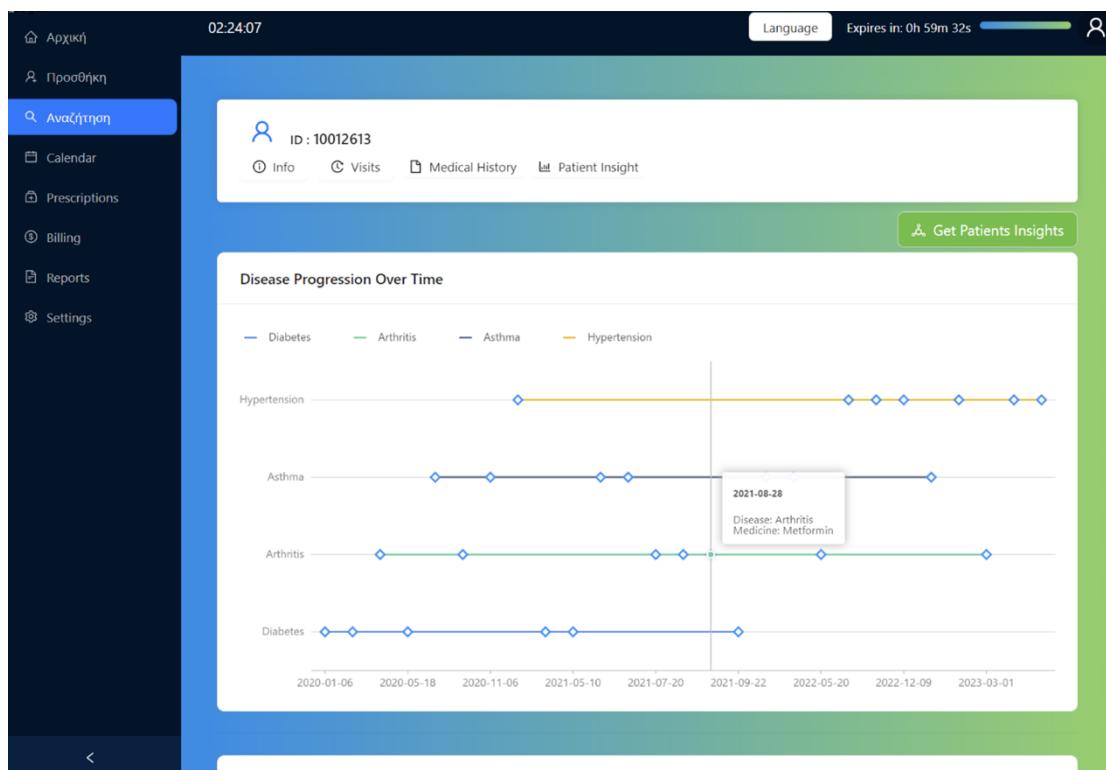
Η συγκεκριμένη αρχιτεκτονική προσφέρει μια σειρά από σημαντικά πλεονεκτήματα:

- Σύνθετες Αναλύσεις Δεδομένων: Επιτρέπει την εκτέλεση σύνθετων αναλύσεων που πηγαίνουν πέρα από τις δυνατότητες της βάσης δεδομένων Cassandra, χάρη στην ισχύ του Apache Spark.
- Προσωποποίηση και Ευελιξία: Προσφέρει τη δυνατότητα για προσωποποιημένες αναλύσεις, με την ενσωμάτωση νέων templates ανάλογα με τις συγκεκριμένες ανάγκες των επαγγελματιών υγείας.
- Δυναμικότητα: Η δυνατότητα για δυναμική παραμετροποίηση και κατ' απαίτηση εκτέλεση των αναλύσεων καθιστά το σύστημα ιδιαίτερα ευέλικτο.
- Έλεγχος Πόρων και Φόρτου: Η αρχιτεκτονική επιτρέπει τον αποτελεσματικό έλεγχο των πόρων, εξασφαλίζοντας ότι οι αναλύσεις διεξάγονται με τον πιο αποδοτικό τρόπο χωρίς να επηρεάζουν την εύρυθμη λειτουργία του συστήματος.

Με αυτόν τον τρόπο, οι επαγγελματίες της υγείας έχουν τη δυνατότητα να προβάλλουν προηγμένες αναλύσεις δεδομένων απευθείας από την εφαρμογή τους όταν απαιτείται, ενισχύοντας έτσι την ποιότητα της φροντίδας που παρέχουν.



Εικόνα 4.65: Spark pipeline in Kubernetes cluster



Εικόνα 4.66: Αποτέλεσμα εκτέλεσης του patientInsights.py (Frontend)

# Κεφάλαιο 5

## ΣΥΜΠΕΡΑΣΜΑΤΑ

### 5.1 Σύνοψη και ανάλυση βασικών σημείων

Στην εποχή της τεχνολογικής εξέλιξης, η υγεία και η ιατρική φροντίδα υποστηρίζονται ολοένα και περισσότερο από τεχνολογικές λύσεις. Τα Πληροφοριακά Συστήματα Υγείας, και ειδικότερα τα συστήματα διαχείρισης Ηλεκτρονικού Φακέλου Υγείας, αποτελούν τον πυρήνα αυτής της ψηφιακής επανάστασης. Η συνεχής εξέλιξη και βελτίωση των συστημάτων ΗΦΥ είναι απαραίτητη για τη διασφάλιση της καλύτερης δυνατής φροντίδας για όλους.

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, αναπτύχθηκε ένα ολοκληρωμένο, παράλληλο και κατανεμημένο σύστημα διαχείρισης Ηλεκτρονικού Φακέλου Υγείας Ασθενούς. Η χρήση τεχνολογιών ανοιχτού κώδικα, όπως το Kubernetes και η βάση δεδομένων Cassandra, επέτρεψε τη δημιουργία ενός πλήρως λειτουργικού και αξιόπιστου συστήματος. Ειδικά, η ενσωμάτωση του Apache Spark αποτέλεσε ένα κρίσιμο στοιχείο της αρχιτεκτονικής, προσφέροντας τη δυνατότητα για γρήγορη επεξεργασία και ανάλυση μεγάλου όγκου δεδομένων.

Η εργασία αυτή δεν είναι απλώς μια μικρογραφία μιας εφαρμογής, αλλά επικεντρώνεται στην ανάπτυξη μιας γενικότερης αρχιτεκτονικής ενός συστήματος διαχείρισης ΗΦΥ. Αναφορικά με την πολυπλοκότητα του σχεδιασμού και της υλοποίησης, ο χρόνος που διατέθηκε για την έρευνα των αναγκών, τον σχεδιασμό της βάσης δεδομένων, τη διεπαφή (backend), το γραφικό περιβάλλον (frontend), καθώς και η ενσωμάτωση στο Kubernetes, ήταν σημαντικός αλλά και απαιτητικός. Παρά την πολυπλοκότητα της σχεδίασης και της υλοποίησης του, το σύστημα που προτείνεται είναι σε θέση να αντιμετωπίσει τις προκλήσεις και τις ανάγκες του τομέα της υγείας.

Με τη χρήση τεχνολογιών ανοιχτού κώδικα, δημιουργήθηκε ένα ολοκληρωμένο και πλήρως λειτουργικό σύστημα. Ένα σύστημα που υπόσχεται αδιάκοπη λειτουργία, επεκτασιμότητα, και κλιμάκωση ανταποκρινόμενο στις μεταβολές του αριθμού των χρηστών. Διαθέτει ευχρηστία και προσαρμοσμένο UI για ασθενείς και επαγγελματίες της υγείας, καθώς και τη δυνατότητα εύκολης προσθήκης λειτουργιών για να καλύψει τις ανάγκες κάθε ειδικότη-

τας. Παρέχει δυνατότητες για ανάλυση δεδομένων υψηλής πολυπλοκότητας και δυναμικής προσθήκης μοντέλων ανάλυσης που εξυπηρετούν την εκάστοτε περίσταση.

Συνοψίζοντας, η εργασία αυτή αποτελεί μια τεχνολογικά προηγμένη, αξιόπιστη και ευέλικτη πρόταση σχεδιασμού και διαχείρισης των Ηλεκτρονικών Φακέλων Υγείας. Η πρόταση αυτή ανταποκρίνεται στις τρέχουσες ανάγκες του τομέα της υγείας και είναι εξοπλισμένη για να ανταπεξέλθει στις μελλοντικές προκλήσεις. Η επέκτασή της είναι δυνατή και κάθε στοιχείο του συστήματος έχει σχεδιαστεί με τέτοιο τρόπο ώστε να το επιτρέπει.

## 5.2 Περιορισμοί έρευνας

Κατά την ανάπτυξη του συστήματος, αντιμετωπίστηκαν ορισμένοι περιορισμοί και δυσκολίες που είναι σημαντικό να αναφερθούν.

Συγκεκριμένα οι περιορισμοί αυτοί περιλαμβάνουν:

**Πόροι Συστήματος:** Το σύστημα είναι πολύπλοκο και απαιτεί σημαντικούς πόρους για να γίνει ενσωμάτωση, συμπεριλαμβανομένης υψηλής υπολογιστικής ισχύος, αποθηκευτικού χώρου και δικτύου.

**Περιορισμένα Πραγματικά Δεδομένα:** Λόγω της έλλειψης πρόσβασης σε πραγματικά δεδομένα, χρησιμοποιήθηκε μηχανή παραγωγής τυχαίων δεδομένων. Αυτό επηρέασε και την ποιότητα των αναλύσεων με τον Spark.

**Χρονικοί Περιορισμοί:** Ο περιορισμένος χρόνος για την έρευνα και την ανάπτυξη του συστήματος επηρέασε το επίπεδο της λεπτομερούς ανάλυσης και δοκίμων του συστήματος. Αυτό μπορεί να οδηγήσει σε λάθη ή αδυναμίες που δεν έχουν εντοπιστεί.

**Τεχνολογικοί Περιορισμοί:** Η ενσωμάτωση μεταξύ διαφορετικών τεχνολογιών, όπως Kubernetes, Cassandra, και Apache Spark, παρουσίασε προκλήσεις στον συντονισμό και την απόδοση.

**Επεκτασιμότητα και Κλιμακωσιμότητα:** Παρόλο που το σύστημα έχει σχεδιαστεί για επεκτασιμότητα, η πραγματική κλιμακωσιμότητα του δεν έχει αξιολογηθεί πλήρως, χρήζοντας περαιτέρω ερευνών για την αξιολόγηση της απόδοσης του.

## 5.3 Μελλοντικές Επεκτάσεις

Ένα τόσο πολύπλοκο σύστημα, όπως το παρόν, ανοίγει τον δρόμο για πολλές μελλοντικές επεκτάσεις και βελτιώσεις. Αναλυτικότερα τα παρακάτω αποτελούν πιθανές επεκτάσεις:

- Ασφάλεια και Κρυπτογράφηση:** Ενσωμάτωση προηγμένων μηχανισμών ασφάλειας και κρυπτογράφησης των δεδομένων, σύμφωνα με τα πρότυπα της Ευρωπαϊκής Ένωσης, όπως το GDPR, και άλλες διεθνείς προδιαγραφές.
- Εξειδίκευση Περιβάλλοντος Χρήστη:** Βελτίωση του περιβάλλοντος χρήστη, ειδικά

για τους επαγγελματίες της υγείας, ώστε να ανταποκρίνεται στις εξειδικευμένες ανάγκες τους.

- **Επέκταση Βάσης Δεδομένων:** Αύξηση της χωρητικότητας και των λειτουργιών της βάσης δεδομένων για να υποστηρίξει περισσότερες εφαρμογές και αναλύσεις.
- **Προσθήκη Συσκευών Live Μετρήσεων:** Ενσωμάτωση συσκευών για live μετρήσεις δεδομένων, όπως έξυπνα ρολόγια και κινητά τηλέφωνα.
- **Εργαλεία Τεχνικής Επίβλεψης και CD/CI:** Ενσωμάτωση εργαλείων για Continuous Deployment και Continuous Integration, προκειμένου να αυτοματοποιηθεί η διαδικασία ανάπτυξης και διανομής του λογισμικού.
- **Επικοινωνία με Άλλα Συστήματα:** Δυνατότητα επικοινωνίας με άλλα συστήματα στον τομέα της υγείας μέσω γνωστών πρωτοκόλλων, όπως το HL7.

# ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] World Health Organization. Διαθέσιμο στο: <https://www.who.int/>
- [2] European Commission. Public health, eHealth: Digital health and care. Διαθέσιμο στο: [https://health.ec.europa.eu/ehealth-digital-health-and-care\\_en](https://health.ec.europa.eu/ehealth-digital-health-and-care_en)
- [3] Σημαιόπουλος Β. (2017). Ηλεκτρονικός Φάκελος Ασθενή. Μεταπτυχιακή Διπλωματική Εργασία. Πρόγραμμα Μεταπτυχιακών Σπουδών: Οικονομικά και Διοίκηση της Υγείας. Πανεπιστήμιο Πειραιά, 2017.
- [4] Carter J.H., (2008). Electronic Health Records, 2nd Edition, American College of Physicians.
- [5] Τσάκωνα, Α. (2014). Καινοτόμος και Θεματικός Ηλεκτρονικός Φάκελος Υγείας Υποστήριξης του Ειδικού Ιατρού στη Πρόγνωση, Διάγνωση και Αντιμετώπιση Ηπατικών Ασθενειών. Διδακτορική Διατριβή. Τμήμα Μηχανικών Η/Υ και Πληροφορικής. Πανεπιστήμιο Πατρών.
- [6] Healthcare Information and Management Systems Society (HIMSS). Διαθέσιμο στο: <https://www.himss.org/>
- [7] Κουμπρούρος, Ι. (2015). Τεχνολογίες Πληροφορίας και Επικοινωνιών στην Υγεία. ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΟΘΗΚΩΝ. Εθνικό Μετσόβιο Πολυτεχνείο. Διαθέσιμο στο [www.kallipos.gr](http://www.kallipos.gr).
- [8] Fossil M., & Dorfman, S. (2013). Electronic health records: strategies for long-term success. Health Administration Press.
- [9] Αποστολοπούλου Μ., (2019). Ατομικός Ηλεκτρονικός Φάκελος: Η στάση των πολιτών απέναντι στην εφαρμογή του στην Ελλάδα. Μεταπτυχιακή διπλωματική εργασία. Διοίκηση Υπηρεσιών Υγείας, Πανεπιστήμιο «Μακεδονία».
- [10] HIMSS. Personal Health Records, Electronic Health Records Key to India's National Digital Health Mission Comment Letter. Διαθέσιμο στο <https://www.himss.org/resources/personal-health-records-electronic-health-records-key-indias-national-digital-health>

- 
- [11] Αγγελίδης, Π. (2015). Ηλεκτρονική Υγεία. ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΟΘΗΚΩΝ. Εθνικό Μετσόβιο Πολυτεχνείο. Διαθέσιμο στο [www.kallipos.gr](http://www.kallipos.gr).
  - [12] HealthIT.gov. Benefits of EHRs. Διαθέσιμο στο: <https://www.healthit.gov/topic/health-it-and-health-information-exchange-basics/benefits-ehrs>
  - [13] Νόμος 4486/2017. Μεταρρύθμιση της Πρωτοβάθμιας Φροντίδας Υγείας, επείγουσες ρυθμίσεις αρμοδιότητας Υπουργείου Υγείας και άλλες διατάξεις. (ΦΕΚ 115/A/7-8-2017).
  - [14] Νόμος 4600/2019. Εκσυγχρονισμός και Αναμόρφωση Θεσμικού Πλαισίου Ιδιωτικών Κλινικών, Σύσταση Εθνικού Οργανισμού Δημόσιας Υγείας, Σύσταση Εθνικού Ινστιτούτου Νεοπλασιών και λοιπές διατάξεις. (ΦΕΚ 43/A/09-03-2019).
  - [15] Οδηγός του πολίτη.(2019). Ηλεκτρονικός φάκελος υγείας και σχετικές ρυθμίσεις. Διαθέσιμο στο: <http://www.odigostoupoliti.eu/atomikos-ilektronikos-fakelos-ygeias-aify/>
  - [16] Σύστημα Πρωτοβάθμιας Φροντίδας Υγείας. Διαθέσιμο στο <https://ehealth.gov.gr/p-rv/p>
  - [17] OpenEMR Project. Διαθέσιμο στο: <http://www.open-emr.org/>
  - [18] OpenEMR Wiki. Διαθέσιμο στο: [https://www.open-emr.org/wiki/index.php/OpenEMR\\_Features](https://www.open-emr.org/wiki/index.php/OpenEMR_Features)
  - [19] GNU Health. Διαθέσιμο στο: <http://health.gnu.org/>
  - [20] GNU Health. WikiBooks. Διαθέσιμο στο: [https://en.wikibooks.org/wiki/GNU\\_Health](https://en.wikibooks.org/wiki/GNU_Health)
  - [21] OpenMRS Project. Διαθέσιμο στο: <https://openmrs.org/>
  - [22] OpenMRS Wiki. Διαθέσιμο στο: <https://wiki.openmrs.org/>
  - [23] Kubernetes. Διαθέσιμο στο: <https://kubernetes.io/>
  - [24] Helm. Διαθέσιμο στο: <https://helm.sh/>
  - [25] Apache Cassandra. Διαθέσιμο στο: <https://cassandra.apache.org/>
  - [26] Apache Spark. Διαθέσιμο στο: <https://spark.apache.org/>
  - [27] React JS. Διαθέσιμο στο: <https://react.dev/>

- 
- [28] Wikipedia. Document Object Model. Διαθέσιμο στο: [https://en.wikipedia.org/wiki/Document\\_Object\\_Model](https://en.wikipedia.org/wiki/Document_Object_Model)
  - [29] <https://www.geeksforgeeks.org/reactjs-virtual-dom/>
  - [30] FastAPI. Διαθέσιμο στο: <https://fastapi.tiangolo.com/>
  - [31] REST API. Διαθέσιμο στο: <https://www.ibm.com/topics/rest-apis>
  - [32] Nginx. Διαθέσιμο στο: <https://www.nginx.com/>
  - [33] ant design reference
  - [34] Ubuntu. Διαθέσιμο στο: <https://ubuntu.com/>
  - [35] K8ssandra. Διαθέσιμο στο: <https://docs.k8ssandra.io/components/k8ssandra-operator/>
  - [36] Datastax. Διαθέσιμο στο: <https://www.datastax.com/what-is/stargate>
  - [37] Github. Διαθέσιμο στο: <https://github.com/bitnami/charts/tree/main/bitnami/spark>

# Παράρτημα Α

## Αρχεία Εφαρμογής

Ο πηγαίος κώδικας της παρούσας διπλωματικής εργασίας βρίσκεται αποθηκευμένος στη σελίδα του GitHub στον παρακάτω σύνδεσμο:

<https://github.com/avouz/Myhealth>