ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΛΙΤΙΚΩΝ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ ΒΑΣΕΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

ΓΕΩΡΓΙΟΣ ΣΟΥΛΤΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Γεώργιος Λιουδάκης
Διδάσκων Π.Δ. 407/80

Λαμία, Οκτώβριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΛΙΤΙΚΩΝ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ ΒΑΣΕΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

ΓΕΩΡΓΙΟΣ ΣΟΥΛΤΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Γεώργιος Λιουδάκης
Διδάσκων Π.Δ. 407/80

Λαμία, Οκτώβριος 2022

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE &
TELECOMMUNICATIONS

DEVELOPMENT OF POLICY ADMINISTRATION POINT FOR ATTRIBUTE BASED ACCESS CONTROL

GEORGIOS SOULTOS

FINAL THESIS

ADVISOR

Georgios Lioudakis
Adjunct Professor

Lamia, October 2022

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 03/11/2022

Ο - Η Δηλ.
Γεώργιος Σούλτος

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Ο έλεγχος πρόσβασης έχει τεράστια σημασία στην ασφάλεια και την ιδιωτικότητα. Πράγματι κάθε παραβίαση στην ασφάλεια αφορά αθέμιτη πρόσβαση. Λόγω της πολυπλοκότητας της δομής των σημερινών οργανισμών/επιχειρήσεων αλλά και της κρισιμότητας των δεδομένων που διαχειρίζονται, κρίνεται απαραίτητη η εφαρμογή ειδικά σχεδιασμένων τεχνικών ελέγχου πρόσβασης για την διασφάλιση της ιδιωτικότητας.

Στην παρούσα πτυχιακή εργασία παρουσιάζεται η ανάπτυξη μιας πλατφόρμας η οποία δίνει την δυνατότητα σε μη τεχνικούς χρήστες να δημιουργούν και να διαχειρίζονται εύκολα πολιτικές ελέγχου πρόσβασης τύπου ABAC χωρίς προαπαιτούμενες γνώσεις πάνω στην Επεκτάσιμη Γλώσσα Σήμανσης Ελέγχου Πρόσβασης (XACML).

ABSTRACT

Access control has great importance in security and privacy fields. Each security violation has to do with inappropriate access to data or resources. Because of the complexity of today's organizations/companies and the importance of the data that they manage, it's critical the application of special designed privacy access control systems to protect privacy.

The final purpose of this thesis is to develop a privacy access control platform, so non-technical users can easily create and manage ABAC policies without any prior knowledge of the Extensible Access Control Markup Language (XACML).

Table of Contents

ΠΕΡΙΛΗΨΗ I
ABSTRACT III

1. ΕΙΣΑΓΩΓΗ 1

2. ΤΕΧΝΟΛΟΓΙΕΣ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ 2

2.1 Τι ειναι ελεγχος προσβασης 2
2.2 DAC 2
2.3 MAC 2
2.4 RBAC 3
2.5 ABAC 4
2.6 XACML 4

3. ΠΡΟΔΙΑΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ 8

3.1 Γενικη περιγραφη του λογισμικου 8
3.2 Απαιτησεις χρηστη 8
3.2.1 Θα τρεχει σε browser, ανεξαρτητως λειτουργικου συστηματος. 8
3.2.2 Θα μπορει να δημιουργει και να φορτωνει project 8
3.2.3 Θα μπορει να φορτωνει αποθηκευμενο project. 8
3.2.4 Θα πρεπει να μπορει να αποθηκευει project. 9
3.2.5 Οι διαθεσιμες κατηγοριες για τα attributes θα ειναι Subject, Resource, Action και Environment. 9
3.2.6 Ο χρηστης θα μπορει να δημιουργει attributes 9
3.2.7 Ο χρηστης θα μπορει να επεξεργαζεται attributes 9
3.2.8 Ο χρηστης θα μπορει να διαγραφει attributes. 9
3.2.9 Θα μπορει να δημιουργει πολλαπλα attribute values για καθε κατηγορια 9
3.2.10 Για καθε κατηγορια θα εμφανιζονται ολα τα attributes που εχει δημιουργησει ο χρηστης 10
3.2.11 Θα μπορει να δημιουργει ABAC policies. 10
3.2.12 Θα μπορει να επεξεργαζεται ABAC policies. 10
3.2.13 Θα μπορει να διαγραφει ABAC policies. 10
3.2.14 Ο χρηστης θα πρεπει να εχει την δυνατοτητα να δει συγκεντρωμενα ολα τα ABAC policies. 10
3.2.15 Θα μπορει να παραγει τον XACML κωδικα για το ABAC Policy. 10
3.2.16 Θα μπορει να παραγει XACML request 11
3.3 Αρχιτεκτονικη συστηματος 11
3.3.1 Διάγραμμα Περιπτώσεων Χρήσης 12
3.3.2 Διάγραμμα Αρχιτεκτονικής 14
3.3.3 Java 11 16
3.3.4 Spring Framework 17
3.3.5 ReactJS 17
3.3.6 API 17
3.3.7 AuthzForce 18
3.3.8 GSON 18
3.3.9 Docker 18
3.3.10 Back-End Service 18

3.3.11 Front-End Service 19
3.3.12 Επικοινωνια με το τοπικο συστημα αρχειων 19

4. ΠΑΡΑΔΕΙΓΜΑ ΧΡΗΣΗΣ 20

4.1 Δημιουργία των attributes και attribute values 20
4.2 Διαχειριση ABAC 23
4.3 Παραγωγη ABAC Request/Policy 27
4.4 Διαχειριση project 30

5. ΣΥΜΠΕΡΑΣΜΑΤΑ 31

6. ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ 31

7. ΒΙΒΛΙΟΓΡΑΦΙΑ 32

Πίνακας Σχημάτων

Σχήμα 1: Αρχιτεκτονική XACML 5
Σχήμα 2: Βασικά στοιχεία XACML 6
Σχήμα 3: Εκτίμηση αποτελέσματος 7
Σχήμα 4: Διάγραμμα περιπτώσεων χρήσης 12
Σχήμα 5: Διάγραμμα αρχιτεκτονικής 14
Σχήμα 6: Βασικές κατηγορίες ABAC 20
Σχήμα 7: Δημιουργία Role attribute 21
Σχήμα 8: Δημιουργία File attribute 21
Σχήμα 9: Δημιουργία Action attribute 22
Σχήμα 10: Δημιουργία Location attribute 22
Σχήμα 11: Προβολή όλων των Subject attributes 23
Σχήμα 12: Επεξεργασία του Role attribute 23
Σχήμα 13: Βασικές ρυθμίσεις πολιτικής 23
Σχήμα 14: Ρυθμίσεις Target για την πολιτική 24
Σχήμα 15: Ρυθμίσεις Rules για την πολιτική 24
Σχήμα 16: Βασικές ρυθμίσεις για το Doctor policy 24
Σχήμα 17: Ρυθμίσεις Target για το doctor policy 24
Σχήμα 18: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα παλιά ιατρικά δεδομένα όσο βρίσκεται εντός του νοσοκομείου 25
Σχήμα 19: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα πρόσφατα ιατρικά δεδομένα όσο βρίσκεται εντός του νοσοκομείου 25
Σχήμα 20: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στις ιδιωτικές του σημειώσεις όσο βρίσκεται εντός του νοσοκομείου 25
Σχήμα 21: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στις συνταγογραφίσεις όσο βρίσκεται εντός του νοσοκομείου 25
Σχήμα 22: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα ραντεβού όσο βρίσκεται εντός του νοσοκομείου 25
Σχήμα 23: Επεξεργασία κανόνων του Policy 26
Σχήμα 24: Προβολή όλων των ABAC policies 26
Σχήμα 25: Επεξεργασία ABAC policy 26
Σχήμα 26: Οθόνη παραγωγής ABAC Policy 27
Σχήμα 27: Οθόνη διαχείρισης του project 30

1. Εισαγωγή

Ο έλεγχος πρόσβασης αποτελεί βασικό και αναπόσπαστο κομμάτι της προστασίας δεδομένων και της ασφάλειας τον υπολογιστικών συστημάτων και των δεδομένων τους. Οι πολυπλοκότητα της δομής των σημερινών οργανισμών/επιχειρήσεων σε συνδυασμό με την διαχείριση δεδομένων κρίσιμης σημασίας, χρίζει απαραίτητη την σωστή εφαρμογή μηχανισμών ελέγχου πρόσβασης για την διαφύλαξη της ασφάλειας και της ιδιωτικότητας των δεδομένων και των υποδομών. Προκειμένου να επιτευκτεί αυτός ο σκοπός έχουν αναπτυχθεί πολλαπλοί μηχανισμοί και μέθοδοι εφαρμογής ελέγχου πρόσβασης: από απλά access control lists που εφαρμόζονται σε λειτουργικά συστήματα και σε απλοϊκές περιπτώσεις μέχρι πολύπλοκους μηχανισμούς όπως Attribute Based Access Control (ABAC) για την διαχείριση του ελέγχου πρόσβασης βάσει χαρακτηριστικών, με ακρίβεια σε πολύπλοκες καταστάσεις. Ωστόσο όσο πιο λεπτομερής και ακριβής είναι η μέθοδος ελέγχου πρόσβασης τόσο πιο δύσκολη γίνεται και η ανάπτυξη του κώδικα XACML που απαιτείται για την εφαρμογή των πολιτικών, ιδιαίτερα από τους μη τεχνικούς χρήστες. Για τον σκοπό αυτό, στόχος αυτής της πτυχιακής είναι η ανάπτυξη μιας πλατφόρμας η οποία δίνει την δυνατότητα σε μη τεχνικούς χρήστες να δημιουργούν και να διαχειρίζονται εύκολα πολιτικές ελέγχου πρόσβασης τύπου ABAC, χωρίς προαπαιτούμενες γνώσεις πάνω στην Επεκτάσιμη Γλώσσα Σήμανσης Ελέγχου Πρόσβασης (XACML).

Στο Κεφάλαιο 2 παρουσιάζονται οι βασικές έννοιες ελέγχου πρόσβασης αλλά και η χρησιμότητα τους στα σύγχρονα υπολογιστικά συστήματα. Επιπλέον γίνεται μία ανασκόπηση στις βασικές μεθόδους ελέγχου πρόσβασης όπως DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role-Based Access Control) και ABAC (Attribute-Based Access Control). Τέλος δίνεται έμφαση στην Επεκτάσιμη Γλώσσα Σήμανσης Ελέγχου Πρόσβασης (XACML) σε συνδυασμό με το ABAC.

Στο Κεφάλαιο 3 γίνεται παρουσίαση του συστήματος διαχείρισης πολιτικών ελέγχου πρόσβασης βάσει χαρακτηριστικών. Αρχικά παρουσιάζεται και αναλύεται το διαγράμματα περιπτώσεων χρήσης αλλά και οι απαιτήσεις του χρήστη. Στην συνέχεια παρουσιάζεται και αναλύεται το διάγραμμα αρχιτεκτονικής του συστήματος. Τέλος γίνεται αναλυτική ανασκόπηση στις τεχνολογίες που χρησιμοποιήθηκαν κατά την ανάπτυξη της πλατφόρμας.

Κλείνοντας στο Κεφάλαιο 4 γίνεται αναπαράσταση της χρήσης του συστήματος μέσω ενός ρεαλιστικού παραδείγματος χρήσης. Πιο συγκεκριμένα το σύστημα χρησιμοποιείται για την δημιουργία και την διαχείριση πολιτικών ελέγχου πρόσβασης σε ένα νοσοκομείο. Έτσι δημιουργώντας τα απαραίτητα χαρακτηριστικά (attributes) αλλά και πολιτικές (policies) οδηγούμαστε στην τελική παραγωγή του XACML κώδικα τόσο για το XACML Request όσο και για το XACML Policy.

2. Τεχνολογίες Ελέγχου Πρόσβασης

2.1 Τι είναι έλεγχος πρόσβασης

Μια σημαντική απαίτηση σε οποιοδήποτε σύστημα πληροφοριών αποτελεί η προστασία των δεδομένων του από κακόβουλη ή μη επιτρεπτή χρήση. Στόχος των τεχνολογιών ελέγχου πρόσβασης αποτελεί η προστασία αυτών των δεδομένων. Ο έλεγχος πρόσβασης αποτελεί μια τεχνική ασφάλειας η οποία η οποία ελέγχει ποιος ή τι και κάτω από ποιες συνθήκες μπορεί να αποκτήσει συγκεκριμένες μορφές πρόσβασης σε ένα υπολογιστικό σύστημα. [0] Τα συστήματα ελέγχου πρόσβασης αποτελούνται από διακριτές πολιτικές, μοντέλα και μηχανισμούς όπου όταν εφαρμόζονται έχουν την δυνατότητα να ελέγξουν και να αποφασίσουν αν οποιαδήποτε επιθυμητή ενέργεια είναι αποδεκτή βάσει των κανόνων που τους έχουν δοθεί. Οι πολιτικές ελέγχου πρόσβασης, αποτελούν υψηλού επιπέδου οδηγίες, οι οποίες καθορίζουν τους τρόπους με τους οποίους ελέγχεται η πρόσβαση και λαμβάνονται οι αποφάσεις πρόσβασης σε ένα υπολογιστικό σύστημα. Ο μηχανισμός ελέγχου πρόσβασης, λειτουργεί σαν ένας επιμελητής αναφοράς όπου μεσολαβεί κάθε προσπάθεια πρόσβασης του χρήστη σε οποιοδήποτε πόρο του συστήματος. Ο επιμελητής αναφοράς συμβουλεύεται τη βάση δεδομένων εξουσιοδοτήσεων για να αποφασίσει εάν ο χρήστης που επιδιώκει να εκτελέσει μία ενέργεια είναι στην πραγματικότητα εξουσιοδοτημένος να την εκτελέσει. Αυτό φυσικά σημαίνει ότι ο έλεγχος πρόσβασης προαπαιτεί την σωστή ταυτοποίηση του χρήστη που επιδιώκει να εκτελέσει κάποια ενέργεια στο σύστημα.

Λόγω της πολυπλοκότητας των σημερινών υπολογιστικών συστημάτων, αλλά και των υψηλών απαιτήσεων ασφάλειας και ιδιωτικότητας που χρειάζονται οι σημερινοί οργανισμοί, η εφαρμογή τεχνολογιών ελέγχου πρόσβασης δεν είναι απλή διαδικασία. Για παράδειγμα πολλές υπηρεσίες δεν χρειάζεται να γνωρίζουν την πραγματική ταυτότητα του χρήστη αλλά μόνο κάποια χαρακτηριστικά ή ιδιότητές του. Αυτές τις ιδιαίτερες απαιτήσεις είναι σε θέση να καλύψουν μέθοδοι ελέγχου πρόσβασης όπως DAC, MAC, RBAC και ABAC.

2.2 DAC

Ο έλεγχος πρόσβασης DAC (Discretionary Access Control) [2] ελέγχει την πρόσβαση των χρηστών πάνω σε πληροφορίες ή υπολογιστικά συστήματα βασισμένος στην ταυτότητα του χρήστη και σε ορισμένους κανόνες εξουσιοδότησης που έχουν οριστεί είτε για μεμονωμένους χρήστες είτε για κάποια ομάδα χρηστών αλλά και για κάθε αντικείμενο μέσα σε ένα σύστημα. Κάθε αίτημα του χρήστη για πρόσβαση σε κάποιο αντικείμενο ενός υπολογιστικού συστήματος, ελέγχεται βάσει των ορισμένων εξουσιοδοτήσεων και εάν υπάρχει έστω και ένας κανόνας ο οποίος επιτρέπει στον συγκεκριμένο χρήστη να αποκτήσει πρόσβαση στο συγκεκριμένο αντικείμενο κάτω από συγκεκριμένες συνθήκες, τότε χορηγείται η πρόσβαση, διαφορετικά η αίτηση για πρόσβαση απορρίπτεται.

2.3 MAC

Παρά το γεγονός ότι η ευελιξία του DAC το καθιστά κατάλληλο για μία μεγάλη ποικιλία συστημάτων και εφαρμογών, ιδιαίτερα σε εμπορικά και βιομηχανικά περιβάλλοντα, η

συγκεκριμένη μέθοδος ελέγχου πρόσβασης υπόκειται στο μειονέκτημα ότι δεν υπάρχει πραγματική ασφάλεια καθώς τα δεδομένα μεταφέρονται σε κάποιο σύστημα καθώς είναι αρκετά εύκολο να παραβιαστούν οι περιορισμοί πρόσβασης μέσω των εξουσιοδοτήσεων. Για παράδειγμα ένας χρήστης που είναι εξουσιοδοτημένος να διαβάζει δεδομένα μπορεί να μεταφέρει τα δεδομένα αυτά σε άλλους χρήστες, οι οποίοι δεν έχουν δικαίωμα να διαβάσουν τα συγκεκριμένα δεδομένα χωρίς την συγκατάθεση του ιδιοκτήτη τους.

Από την άλλη μεριά οι πολιτικές ελέγχου πρόσβασης τύπου MAC (Mandatory Access Control) [2] εμποδίζουν την διάδοση των δεδομένων από αντικείμενα υψηλού επιπέδου σε αντικείμενα χαμηλού επιπέδου. Κάθε υποκείμενο και κάθε αντικείμενο έχει ανατεθεί σε μία κλάση πρόσβασης. Το σύνολο των κλάσεων στις περισσότερες περιπτώσεις αποτελείται από ένα επίπεδο ασφαλείας και ένα σύνολο από κατηγορίες. Το επίπεδο ασφαλείας είναι ένα ταξινομημένο σύνολο το οποίο καθορίζει την ευαισθησία της πληροφορίας. Κάθε επίπεδο ασφαλείας ελέγχει τον εαυτό του και όλα τα επίπεδα κάτω από αυτό. Το σύνολο των κατηγοριών είναι ένα μη ταξινομημένο σύνολο όπου τα στοιχεία του αντιπροσωπεύουν την λειτουργία ή την αρμοδιότητα των περιοχών που μπορούν να δράσουν (πχ. οικονομικά, δημογραφικά, ιατρικά, κλπ.). Έτσι προκειμένου να χορηγηθεί πρόσβαση από μία κλάση πρόσβασης σε μία άλλη, αρχικά γίνεται έλεγχος στο επίπεδο ασφαλείας των κλάσεων και στην συνέχεια ελέγχονται οι ταυτόσημες κατηγορίες προκειμένου το σύστημα να βεβαιωθεί ότι οι δύο κλάσεις έχουν δικαίωμα πρόσβασης στις ίδιες περιοχές.

2.4 RBAC

Ο έλεγχος πρόσβασης RBAC (Role-Based Access Control) [3] ρυθμίζει την πρόσβαση στους χρήστες ενός συστήματος βάσει των ρόλων που τους έχουν δοθεί. Οι συγκεκριμένες πολιτικές απαιτούν την ταυτοποίηση των ρόλων μέσα σε ένα σύστημα. Ένα ρόλος μπορεί να οριστεί σαν ένα σύνολο από ενέργειες και ευθύνες οι οποίες έχουν συσχετιστεί με μία συγκεκριμένη δραστηριότητα. Έτσι αντί να χρειάζεται να χορηγηθούν ξεχωριστά όλα τα διαφορετικά δικαιώματα πρόσβασης σε κάθε χρήστη, τα δικαιώματα πρόσβασης ορίζονται μέσα από τους ρόλους τους. Με αυτό τον τρόπο απλοποιείται σε μεγάλο βαθμό η διαδικασία της εξουσιοδότησης. Ένα ακόμη πλεονέκτημα του RBAC αποτελεί το γεγονός ότι οι ρόλοι επιτρέπουν στους χρήστες να χρησιμοποιούν ένα υπολογιστικό σύστημα με τα λιγότερα δυνατά δικαιώματα πρόσβασης (least privilege). Με άλλα λόγια σε κάθε χρήστη έχουν χορηγηθεί μόνο όσα δικαιώματα είναι απαραίτητα και τίποτα επιπλέον. Έτσι ελαχιστοποιείται ο κίνδυνος ακούσιων σφαλμάτων και ταυτόχρονα περιορίζονται οι δυνατότητες των κακόβουλων χρηστών να εκμεταλλευτούν το σύστημα. Αξίζει να σημειωθεί ότι σε ορισμένες περιπτώσεις μπορούν να αποδοθούν διαφορετικοί ρόλοι σε κάποιους χρήστες. Επιπλέον ένας ρόλος μπορεί να έχει δοθεί σε πολλαπλούς χρήστες ταυτόχρονα. Αυτό συμβαίνει γιατί σε ορισμένα πρότυπα ελέγχου πρόσβασης βάσει ρόλων, ένας χρήστης έχει το δικαίωμα να ασκεί ενέργειες που αντιστοιχούν σε πολλαπλούς ρόλους ταυτόχρονα. Από την άλλη μεριά, κάποια άλλα πρότυπα δεν επιτρέπουν την ταυτόχρονη αντιστοίχιση πολλαπλών ρόλων στον ίδιο χρήστη.

2.5 ABAC

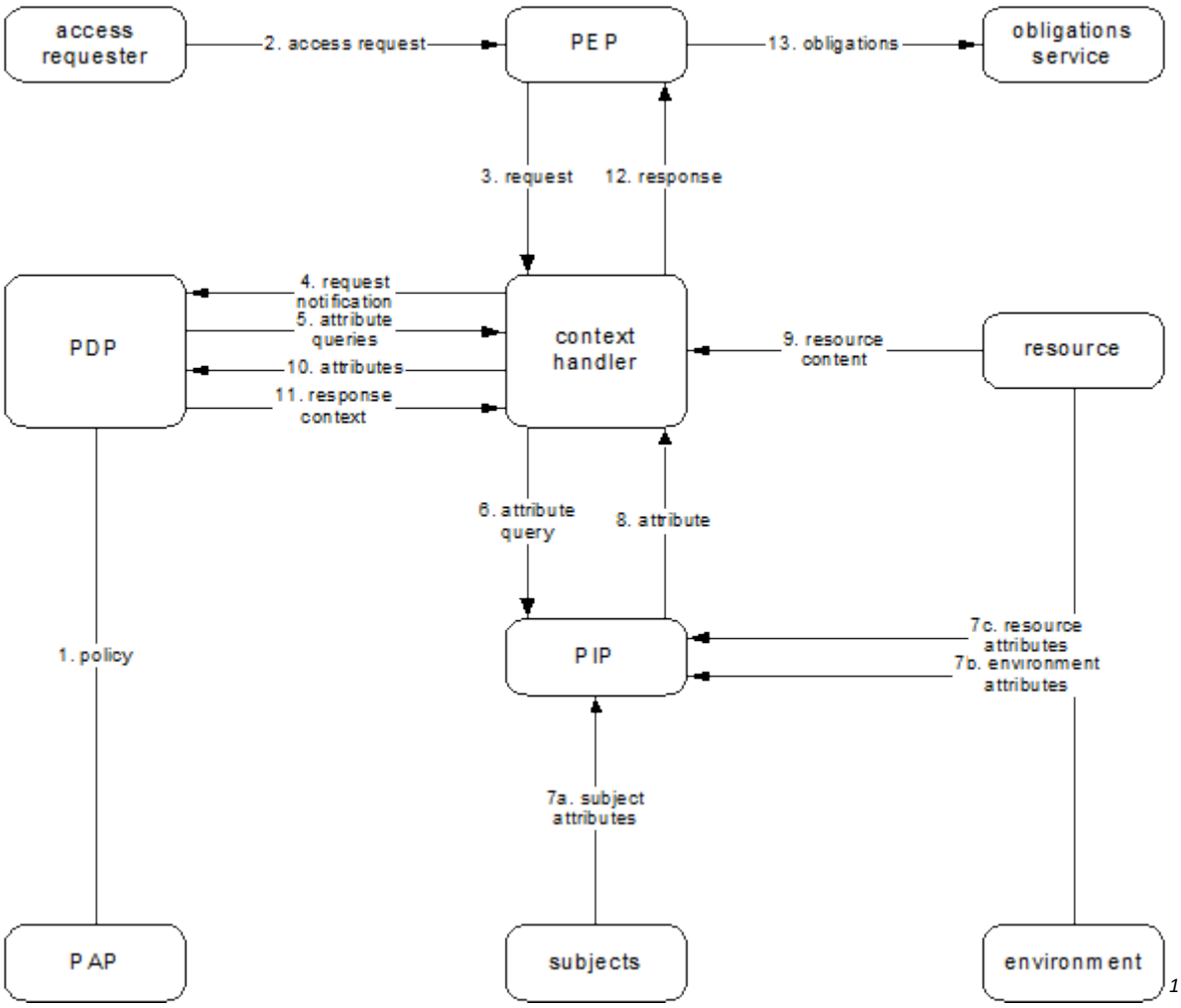
Στα σημερινά κατανεμημένα συστήματα, τα οποία χαρακτηρίζονται από ανεξάρτητους εξυπηρετητές (servers) οι οποίοι προσφέρουν υπηρεσίες σε οποιονδήποτε τις χρειάζεται, οι παραδοσιακοί τρόποι ελέγχου πρόσβασης δεν είναι σε θέση να καλύψουν πλήρως τις ανάγκες αυτών των συστημάτων. Με άλλα λόγια, καθώς τα αιτήματα πρόσβασης μπορεί να προέρχονται ακόμα και από άγνωστους χρήστες τα συστήματα ελέγχου πρόσβασης που βασίζονται στην ταυτότητα του χρήστη μπορούν να εφαρμοστούν.

Μία εναλλακτική λύση σε αυτό το πρόβλημα αποτελεί ο έλεγχος πρόσβασης βάσει χαρακτηριστικών (Attribute-Based Access Control). Το ABAC [4] είναι ένα μοντέλο εξουσιοδότησης το οποίο βασίζεται στα χαρακτηριστικά των χρηστών αντί για τους ρόλους τους. Η βασική ιδέα είναι ότι οι αποφάσεις του ελέγχου πρόσβασης δεν βασίζονται πλέον μόνο στην ταυτότητα του χρήστη αλλά και στα χαρακτηριστικά του. Για παράδειγμα μία πληροφορία σχετικά με τον ρόλο ενός χρήστη (πχ. γιατρός) ή η ημερομηνία γέννησής του μπορεί να είναι πιο σημαντική από την ίδια την ταυτότητα του χρήστη κατά την διαδικασία ελέγχου πρόσβασης.

2.6 XACML

Η XACML (eXtensible Access Control Markup Language) [5] είναι ένα πρότυπο του OASIS που βασίζεται στην γλώσσα XML και αποτελεί μια γλώσσα περιγραφής προδιαγραφών ελέγχου πρόσβασης. Έχει αναπτυχθεί για να υποστηρίζει τους περισσότερους μηχανισμούς πολιτικής ασφαλείας και χρησιμοποιείται ήδη από πολλούς οργανισμούς και εταιρίες. Επιπλέον παρέχει βασικά σημεία επέκτασης για τον ορισμός καινούριων συναρτήσεων, τύπων δεδομένων, συνδυασμό πολλαπλών πολιτικών, καθιστώντας την έτσι ιδανική για την διαχείριση τον προδιαγραφών ελέγχου πρόσβασης.

Επιπλέον υποστηρίζει πλήρως πολιτικές ελέγχου πρόσβασης τύπου ABAC και η εκτίμηση των αποτελεσμάτων μπορεί να γίνει με επιπλέον δεδομένα τα οποία παρέχονται από το Policy Information Point (PIP) το οποίο ορίζεται από την XACML.



Σχήμα 1: Αρχιτεκτονική XACML

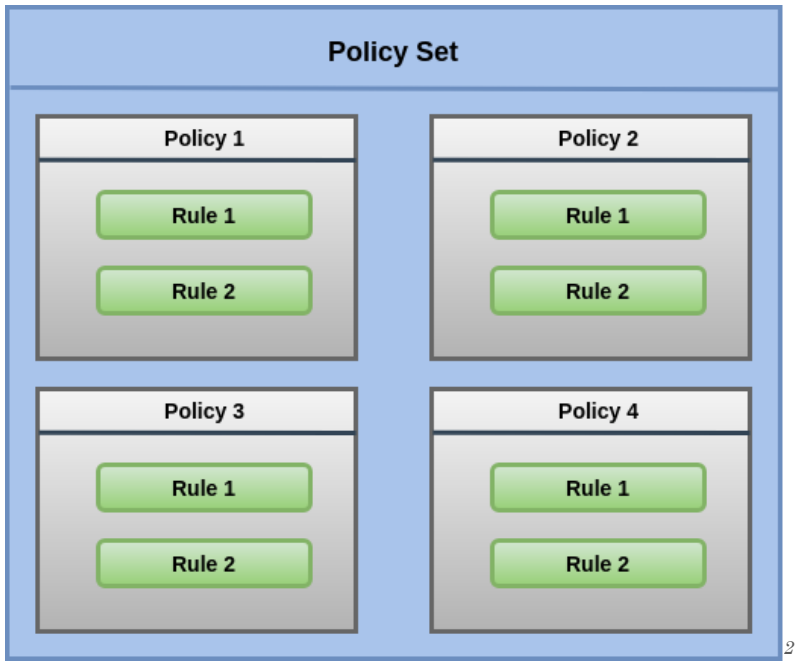
Το Policy Decision Point (PDP) εκτιμά τις πολιτικές βάσει αιτήσεων (requests) που παρέχονται από το Policy Enforcement Points (PEP). Προκειμένου να υπολογιστεί η τελική απόφαση, το PDP είναι πιθανό να χρησιμοποιήσει δεδομένα από το Policy Information Point (PIP) προκειμένου να συλλέξει πληροφορίες όπως χαρακτηριστικά (attributes) σχετικά με τον χρήστη ή οποιοδήποτε άλλο χαρακτηριστικό (attribute) χρειάζεται το αίτημα (request).

Η σύνταξη της γλώσσας XACML στον πυρήνα της αποτελείται από τρία κομμάτια:

1. **<Rule>**
   Περιέχει μία λογική έκφραση η οποία εκτιμάται απομονωμένα χωρίς να καθορίζει την τελική απόφαση από μόνη της.
2. **<Policy>**
   Περιέχει ένα σύνολο από στοιχεία <Rules> και έναν αλγόριθμο για τον συνδυασμό τον αποτελεσμάτων της εκτίμησης. Αποτελεί βασικό κομμάτι της πολιτικής το οποίο χρησιμοποιείται από το PDP για τον υπολογισμό της τελικής απόφασης.

[1] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

3. **<PolicySet>**
   Περιέχει ένα σύνολο από στοιχεία <Policy> ή <PolicySet> και ορίζει τον αλγόριθμο για τον συνδυασμό των αποτελεσμάτων της εκτίμησης.



[2]

Σχήμα 2: Βασικά στοιχεία XACML

Καθένα από τα παραπάνω στοιχεία της XACML περιέχει ένα στοιχείο <Target> το οποίο καθορίζει τις περιστάσεις κατά τις οποίες το συγκεκριμένο <Rule>, <Policy> ή <PolicySet> εφαρμόζεται. Ένα καινό <Target> εφαρμόζεται σε κάθε αίτημα (request), διαφορετικά η εφαρμογή γίνεται βάσει των στοιχείων <AnyOf>, <AllOf> και <Match>.

[2] https://medium.com/identity-beyond-borders/a-beginners-guide-to-xacml-6dc75b547d55

<AnyOf> values	Target value
All "Match"	"Match"
At least one "No Match"	"No Match"
Otherwise	Indeterminate

<AllOf> values	<AnyOf> value
At least one "Match"	"Match"
None matches & at least one "Indeterminate"	"Indeterminate"
All "No Match"	"No Match"

<Match> values	<AllOf> value
All "True"	"Match"
No "False" & at least one "Indeterminate"	"Indeterminate"
At least one "False"	"No Match"

[3]

Σχήμα 3: Εκτίμηση αποτελέσματος

Ένα επιπλέον πεδίο που υποστηρίζει η γλώσσα XACML είναι το rule-combining-algorithm το οποίο ορίζει τον αλγόριθμο με τον οποίο συνδυάζονται τα αποτελέσματα της εκτίμησης των στοιχείων <Rule> με τα αποτελέσματα της εκτίμησης των στοιχείων <Policy>. Μερικοί από τους πιο συνηθισμένους αλγορίθμους είναι οι εξής:

- **Deny-overrides** (*urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides*)
  Αν η απόφαση είναι "Άρνηση" τότε το αποτέλεσμα θα είναι "Άρνηση".
- **Permit-overrides** (*urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides*)
  Αν η απόφαση είναι "Αποδοχή" τότε το αποτέλεσμα θα είναι "Αποδοχή".
- **First-applicable** (*urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable*)
  Σε αυτή την περίπτωση κάθε κανόνας (<Rule>) εκτιμάται με την σειρά που έχει οριστεί μέσα στην πολιτική (<Policy>). Για κάθε κανόνα (<Rule>) αν ο στόχος του (<Target>) ταιριάζει με την συνθήκη, τότε το αποτέλεσμα θα είναι το αποτέλεσμα της συνθήκης, διαφορετικά ελέγχεται ο επόμενος κανόνας (<Rule>).

Επιπλέον η γλώσσα XACML είναι δομημένη γύρω από τις βασικές κατηγορίες ABAC δηλαδή:

- **Subject** (*urn:oasis:names:tc:xacml:3.0:attribute-category:subject*)
- **Resource** (*urn:oasis:names:tc:xacml:3.0:attribute-category:resource*)
- **Action** (*urn:oasis:names:tc:xacml:3.0:attribute-category:action*)

[3] https://medium.com/identity-beyond-borders/a-beginners-guide-to-xacml-6dc75b547d55

- **Environment** (*urn:oasis:names:tc:xacml:3.0:attribute-category:environment*)

Τέλος αξίζει να σημειωθεί πως από την έκδοση XACML 3.0 και μετά υποστηρίζονται και προσαρμοσμένες κατηγορίες ABAC.

3. Προδιαγραφή συστήματος

3.1 Γενική περιγραφή του λογισμικού

Στόχος της πλατφόρμας είναι η εύκολη δημιουργία και διαχείριση των ABAC Policies/Requests. Μέσα από το περιβάλλον διεπαφής, ο χρήστης έχει την δυνατότητα να δημιουργήσει και να επεξεργαστεί attributes για όλες τις βασικές κατηγορίες του ABAC (Subject, Resource, Action, Environment). Επιπλέον του δίνετε η δυνατότητα να δημιουργήσει ABAC πολιτικές και να παράξει τόσο ABAC Policies σε γλώσσα XACML, όσο και XACML Requests. Προκειμένου ο χρήστης να μπορεί να οργανώσει και να μοιραστεί την δουλεία του, η πλατφόρμα προσφέρει δυνατότητα αποθήκευσης και φόρτωσης project σε μορφή JSON. Με αυτό τον τρόπο, ακόμη και ένας μη εξοικειωμένος με το ABAC χρήστης, μπορεί να διαχειρίζεται όλες τις πολιτικές ελέγχου πρόσβασης χωρίς να χρειάζεται να είναι εξοικειωμένος με την γλώσσα XACML.

3.2 Απαιτήσεις χρήστη

3.2.1 Θα τρέχει σε browser, ανεξαρτήτως λειτουργικού συστήματος.

Προκειμένου η πλατφόρμα να υποστηρίζεται εύκολα από όλα τα λειτουργικά συστήματα, θα πρέπει να υποστηρίζεται από όλους τους σύγχρονους περιηγητές ιστού. Έτσι το Front-End Service, το οποίο είναι υπεύθυνο για το περιβάλλον διεπαφής του χρήστη θα πρέπει να έχει φτιαχτεί με τρόπο ώστε να εξυπηρετεί όλους τους εργονομικούς σκοπούς του χρήστη ανάλογα με την συσκευή που χρησιμοποιεί.

3.2.2 Θα μπορεί να δημιουργεί και να φορτώνει project.

Θα δίνεται η δυνατότητα αποθήκευσης και φόρτωσης project. Με άλλα λόγια ο χρήστης θα έχει την δυνατότητα να αποθηκεύσει την δουλειά του σε μορφή αρχείου JSON και να ξαναφορτώσει το αποθηκευμένο αρχείο όποτε επιθυμεί. Έτσι γίνεται απλούστερη η οργάνωση των διαφορετικών πολιτικών και ταυτόχρονα γίνεται ευκολότερος ο διαμοιρασμός των projects.

3.2.3 Θα μπορεί να φορτώνει αποθηκευμένο project.

Στην αρχική οθόνη, ο χρήστης θα έχει την επιλογή να φορτώσει κάποιο ήδη αποθηκευμένο project. Σε αυτή την επιλογή θα εμφανίζεται παράθυρο στο οποίο θα πρέπει να επιλέξει το αρχείο του ήδη αποθηκευμένου project. Στην συνέχεια η εφαρμογή θα διαβάζει το αρχείο και εφόσον είναι έγκυρο, θα φορτώνει τα δεδομένα του στην εφαρμογή.

3.2.4 Θα πρέπει να μπορεί να αποθηκεύει project.

Καθ' όλη την διάρκεια της επεξεργασίας του project, ο χρήστης θα έχει την δυνατότητα να αποθηκεύσει τοπικά τις αλλαγές που έχει κάνει. Για τον λόγο αυτό στην αρχική οθόνη ο χρήστης θα έχει την επιλογή να αποθηκεύσει τοπικά τις αλλαγές του σε μορφή JSON αρχείου.

3.2.5 Οι διαθέσιμες κατηγορίες για τα attributes θα είναι Subject, Resource, Action και Environment.

Στο πάνω μέρος της οθόνης θα υπάρχει ένα μενού με όλες τις διαθέσιμες ABAC κατηγορίες (Subject, Resource, Action, Environment). Το εργονομικό αυτό μενού θα δίνει την δυνατότητα στον χρήστη να προβάλει, να επεξεργαστεί ή να διαγράψει attributes για οποιαδήποτε από τις παραπάνω κατηγορίες επιθυμεί.

3.2.6 Ο χρήστης θα μπορεί να δημιουργεί attributes.

Επιλέγοντας κάποια από τις διαθέσιμες κατηγορίες, ο χρήστης θα έχει την δυνατότητα να δημιουργήσει κάποιο καινούριο attribute για την συγκεκριμένη κατηγορία. Πατώντας λοιπόν το κουμπί προσθήκης attribute θα εμφανίζεται μία φόρμα σε μορφή pop-up window, στην οποία θα μπορεί να συμπληρώσει τις απαραίτητες πληροφορίες για το attribute που θέλει να δημιουργήσει, καθώς και να προσθέσει πολλαπλά attribute values.

3.2.7 Ο χρήστης θα μπορεί να επεξεργάζεται attributes.

Επιλέγοντας κάποια από τις διαθέσιμες κατηγορίες και πατώντας πάνω σε κάποιο attribute, ο χρήστης θα έχει την δυνατότητα να το επεξεργαστεί. Θα εμφανίζεται η ίδια φόρμα που εμφανίζεται και στην δημιουργία του attribute και θα του δίνεται η δυνατότητα να τροποποιήσει όλες τις πληροφορίες του.

3.2.8 Ο χρήστης θα μπορεί να διαγράφει attributes.

Επιλέγοντας κάποια από τις διαθέσιμες κατηγορίες και πατώντας πάνω σε κάποιο attribute, ο χρήστης θα έχει την δυνατότητα να προβεί στην διαγραφή του. Αρχικά θα εμφανίζεται το pop-up window επεξεργασίας του attribute και πατώντας πάνω στο κουμπί διαγραφής θα εμφανίζεται ένα μήνυμα επιβεβαίωσης και μόνο όταν ο χρήστης απαντήσει θετικά το attribute θα διαγράφεται.

3.2.9 Θα μπορεί να δημιουργεί πολλαπλά attribute values για κάθε κατηγορία.

Στην οθόνη δημιουργίας attribute, θα μπορεί να δημιουργήσει αυτόματα πολλαπλά attribute values, διαχωρίζοντάς τα με (,).

3.2.10 Για κάθε κατηγορία θα εμφανίζονται όλα τα attributes που έχει δημιουργήσει ο χρήστης

Κάνοντας κλικ πάνω σε κάποια κατηγορία attribute θα εμφανίζονται όλα τα attributes που έχει δημιουργήσει ο χρήστης για την συγκεκριμένη κατηγορία. Τα attributes θα εμφανίζονται μέσα σε έναν πίνακα, το ένα κάτω από το άλλο, και ο χρήστης θα μπορεί να τα ταξινομήσει βάσει των στηλών του πίνακα.

3.2.11 Θα μπορεί να δημιουργεί ABAC policies.

Στο μενού, στο πάνω μέρος της οθόνης, θα υπάρχει η επιλογή ABAC. Κάνοντας κλικ θα υπάρχει επιλογή για την δημιουργία καινούριο ABAC policy. Στην συνέχεια θα εμφανίζεται η οθόνη δημιουργίας του policy. Εκεί ο χρήστης, θα μπορεί να ρυθμίσει το description, το policy ID, το rule combining algorithm και το max delegation depth. Στην συνέχεια, για κάθε κατηγορία attribute θα επιλέγει τα attribute values για τα οποία θέλει να ισχύει το συγκεκριμένο policy. Τέλος θα μπορεί να προσθέσει κανόνες για κάθε κατηγορία attribute και να επιλέξει την τελική απόφαση του κάθε κανόνα από ένα dropdown menu, με τις επιλογές permit και deny.

3.2.12 Θα μπορεί να επεξεργάζεται ABAC policies.

Πατώντας πάνω σε κάποιο ABAC policy, θα υπάρχει η επιλογή για επεξεργασία συγκεκριμένου policy. Έτσι θα εμφανίζεται η οθόνη δημιουργίας του ABAC policy, και ο χρήστης θα έχει την δυνατότητα να αλλάξει όλες τις ρυθμίσεις και τις συνθήκες του συγκεκριμένου κανόνα, προσθέτοντας και αφαιρώντας attribute values, αλλά και να μεταβάλει την τελική απόφαση του κανόνα.

3.2.13 Θα μπορεί να διαγράφει ABAC policies.

Πατώντας πάνω σε κάποιο ABAC policy, θα υπάρχει η επιλογή “διαγραφή” του συγκεκριμένου policy. Θα εμφανίζεται η οθόνη δημιουργίας του ABAC policy και πατώντας πάνω στο κουμπί διαγραφής θα εμφανίζεται μήνυμα επιβεβαίωσης της διαγραφής του, και μόνο όταν ο χρήστης απαντήσει θετικά θα διαγράφεται το ABAC policy.

3.2.14 Ο χρήστης θα πρέπει να έχει την δυνατότητα να δει συγκεντρωμένα όλα τα ABAC policies.

Κάνοντας κλικ πάνω στην επιλογή ABAC policies στο μενού, στο πάνω μέρος της οθόνης, θα εμφανίζονται συγκεντρωμένα όλα τα ABAC policies που έχει δημιουργήσει ο χρήστης. Τα ABAC policies θα εμφανίζονται μέσα σε ένα πίνακα, το ένα κάτω από το άλλο, και ο χρήστης θα μπορεί να τα ταξινομήσει βάσει των στηλών του πίνακα.

3.2.15 Θα μπορεί να παράγει τον XACML κώδικα για το ABAC Policy.

Αφού έχει δημιουργηθεί κάποιο ABAC policy, ο χρήστης θα έχει την δυνατότητα να παράξει XACML κώδικα για το συγκεκριμένο policy. Έτσι στην αρχική οθόνη θα υπάρχει

ένα drop-down menu από το οποίο θα μπορεί να επιλέξει το ABAC policy για το οποίο θέλει να παράξει τον XACML κώδικα, και στην συνέχεια πατώντας το κουμπί “Generate ABAC Policy” θα μπορεί να παράξει και να αποθηκεύσει το XML αρχείο το οποίο θα περιέχει το XACML κώδικα, τοπικά στην συσκευή του.

3.2.16 Θα μπορεί να παράγει XACML request

Αφού έχουν δημιουργηθεί τα attributes για κάποιες ή όλες από τις διαθέσιμες κατηγορίες, ο χρήστης θα έχει την δυνατότητα να παράξει XACML request για αυτά τα attributes. Έτσι στην αρχική οθόνη θα υπάρχει επιλογή “Generate XACML Request”, όπου θα παράγει τον XACML κώδικα και θα μπορεί να αποθηκεύσει το XML αρχείο τοπικά στην συσκευή του.

3.3 Αρχιτεκτονική συστήματος

Το policy tool, αποτελείται από δύο κομμάτια. Το Front-End και το Back-End. Το Front-End αποτελεί το περιβάλλον διεπαφής του χρήστη, μέσα από το οποίο του δίνεται η δυνατότητα να δημιουργήσει και να διαχειριστεί τους κανόνες ελέγχου πρόσβασης. Το Back-End τρέχει το API για την διαχείριση των λειτουργιών της πλατφόρμας.

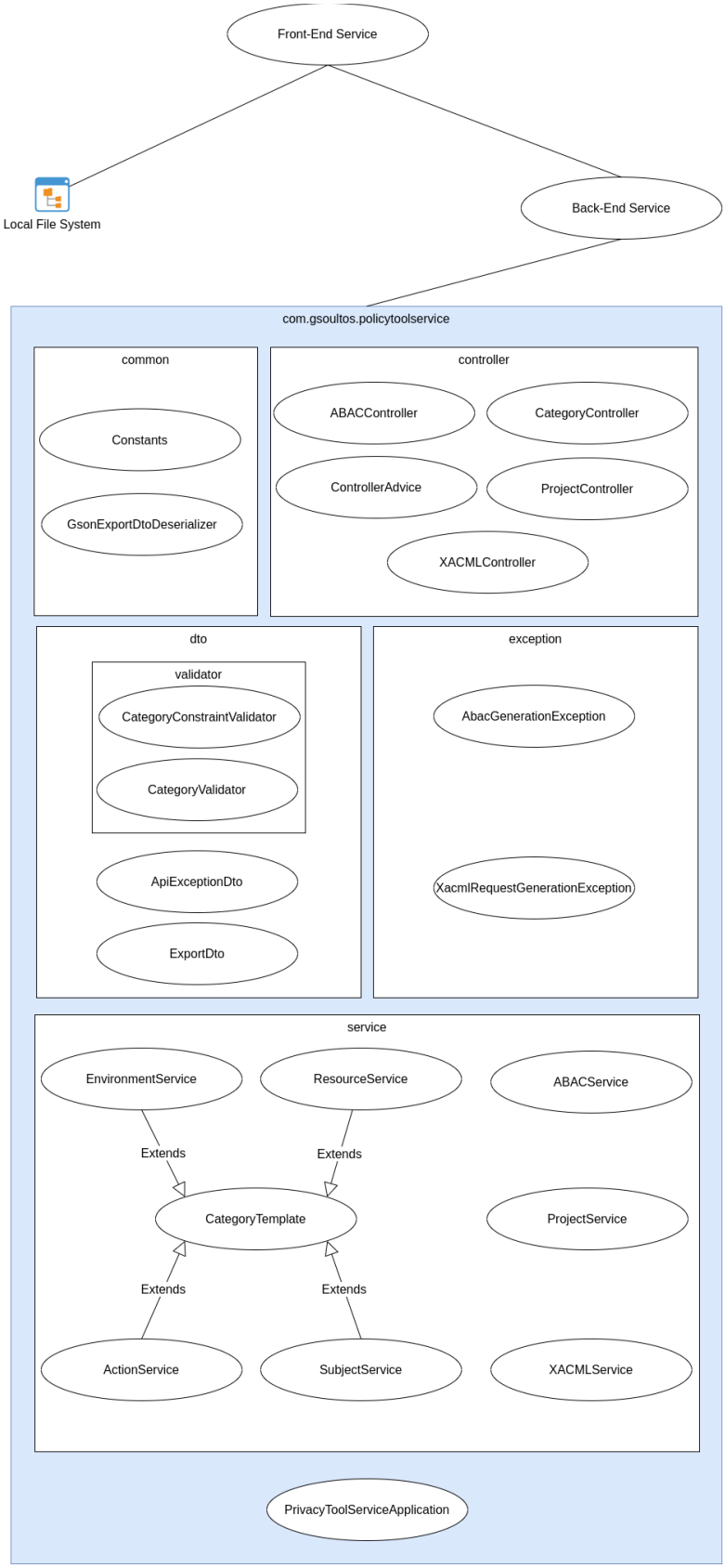
Υπάρχει άμεση επικοινωνία μεταξύ του Front-End και του Back-End. Κάθε εντολή που δίνει ο χρήστης στο Front-End ουσιαστικά αποτελεί μία κλήση σε κάποιο API endpoint του Back-End service. Όλα τα requests από το Front-End στο Back-End γίνονται με την χρήση του HTTP πρωτοκόλλου.

3.3.1 Διάγραμμα Περιπτώσεων Χρήσης

Σχήμα 4: Διάγραμμα περιπτώσεων χρήσης

Στο παραπάνω διάγραμμα αναγράφονται οι δυνατότητες της πλατφρόμας. Όπως φαίνεται ο χρήστης έχει την δυνατότητα να ασκήσει πολλαπλές ενέργειες, όπως Show Attributes, Add Attribute, Edit Attribute, Delete Attribute πάνω στις βασικές κατηγορίες ABAC. Οι ίδιες ενέργειες μπορούν να εκτελεστούν και για τη διαχείριση των ABAC Policies. Όσο αφορά την διαχείριση του project, δίνεται η δυνατότητα αποθήκευσης και φόρτωσης των δεδομένων της πλατφόρμας μέσω επικοινωνίας της με τοπικό σύστημα αρχείων. Τέλος, η παραγωγή των ABAC Policies σε γλώσσα XACML αλλά και η παραγωγή του XACML Request λαμβάνει χώρα μέσω της επικοινωνίας της πλατφόρμας με την βιβλιοθήκη AuthzForce Java API.

3.3.2 Διάγραμμα Αρχιτεκτονικής



Σχήμα 5: Διάγραμμα αρχιτεκτονικής

Το παραπάνω διάγραμμα περιγράφει την αρχιτεκτονική του συστήματος. Όπως φαίνεται η πλατφόρμα αποτελείται από δύο κομμάτια. Το Front-End Service[4], το οποίο τρέχει το περιβάλλον διεπαφής του χρήστη, και το Back-End Service[5], το οποίο τρέχει τον αλγόριθμο παραγωγής των ABAC Policies, XACML Requests αλλά και της διαχείρισης του project. Επιπλέον υπάρχει άμεση επικοινωνία μεταξύ του Front-End Service και του τοπικού συστήματος αρχείων, προκειμένου η πλατφόρμα να μπορεί να αποθήκευση αλλά και φορτώσει δεδομένα από την συσκευή του χρήστη. Η επικοινωνία του Front-End Service με το Back-End Service γίνεται μέσω HTTP Requests, αφού το Back-End Service τρέχει REST API.

Όσο αφορά το Back-End Service έχουν αναπτυχθεί τα ακόλουθα πακέτα στην γλώσσα Java.

- **common**
  Το πακέτο common, περιέχει κλάσεις οι οποίες χρησιμοποιούνται σε πολλαπλά σημεία του κώδικα. Συγκεκριμένα το πακέτο αυτό αποτελείται από τις κλάσεις Constants και GsonExportDtoDeserializer. Η κλάση Constants περιέχει σταθερά στατικά δεδομένα τα οποία χρησιμοποιούνται σε πολλαπλές κλάσεις μέσα στην πλατφόρμα. Η κλάση GsonExportDtoDeserializer είναι μία βοηθητική κλάση για την βιβλιοθήκη GSON, προκειμένου να διεξαχθεί ομαλά η διαδικασία deserialization των προσαρμοσμένων τύπων δεδομένων της πλατφόρμας, έτσι ώστε να μπορέσει να παραχθεί το JSON αρχείο για την τοπική αποθήκευση των δεδομένων του project.
- **controller**
  Το πακέτο controller, περιέχει όλες τις κλάσεις που υλοποιούν κάποιο Spring Controller. Με άλλα λόγια περιέχει τις κλάσεις οι οποίες ορίζουν τα HTTP endpoints του Back-End API. Πιο συγκεκριμένα οι κλάσεις ABACController, CategoryController, ProjectController και XACMLController, αποτελούν κλάσεις οι οποίες ορίζουν τα HTTP endpoints και όλες μαζί συντελούν στην δημιουργία ενός ολοκληρωμένου REST API. Η κλάση ControllerAdvice ορίζει τεχνικές διαχείρισης σφαλμάτων για διαφορετικούς τύπους εξαιρέσεων που μπορεί να δημιουργηθούν καθ' όλης την διάρκεια εκτέλεσης των Controller.
- **dto**
  Το πακέτο dto, περιέχει κλάσεις οι οποίες υλοποιούν DTO (Data Transfer Object), κλάσεις δηλαδή που χρησιμοποιούνται για την μεταφορά δεδομένων κυρίως μεταξύ του Back-End Service και του Front-End Service. Έτσι η κλάση ApiExceptionDto χρησιμοποιείται κατά την αποστολή σφαλμάτων του API από το Back-End στο Front-End. Για παράδειγμα στην περίπτωση που ο χρήστης προσπαθήσει να διαγράψει ένα attribute το οποίο δεν υπάρχει, το API μέσω της κλάσης ApiExceptionDto θα στείλει το timestamp, τον HTTP κωδικό σφάλματος, την HTTP περιγραφή του σφάλματος και τέλος ένα φιλικό προς τον χρήστη μήνυμα σφάλματος. Η κλάση ExportDto περιέχει πεδία για την αποθήκευση των δεδομένων της πλατφόρμας κατά την διάρκεια παραγωγής του αρχείου JSON, το οποίο χρησιμοποιείται για την εξαγωγή και τοπική αποθήκευση του project. Μέσα στο πακέτο dto, υπάρχει εμφωλευμένο το πακέτο validator, το οποίο περιέχει

[4] https://github.com/gsoultos/policy-tool
[5] https://github.com/gsoultos/policy-tool-service

κλάσεις οι οποίες υλοποιούν input validation και χρησιμοποιούνται για την επικύρωση των δεδομένων που λαμβάνονται μέσω των API endpoints. Πιο συγκεκριμένα το interface CategoryValidator υλοποιεί ένα καινούριο annotation για το validaton των διαθέσιμων κατηγοριών ABAC και σε συνδυασμό με την κλάση CategoryConstraintValidator η οποία υλοποιεί την κλάση ConstraintValidator συντελούν στην δημιουργία ενός προσαρμοσμένου annotation το οποίο χρησιμοποιείται κατά κύριο λόγο στα API endpoints τα οποία λαμβάνουν σαν παράμετρο κάποια από τις διαθέσιμες κατηγορίες ABAC.

- **exception**
  Το πακέτο exception, περιέχει κλάσεις που υλοποιούν υποκλάσεις τύπου Exception και αποτελούν προσαρμοσμένους τύπους σφαλμάτων. Έτσι η κλάση AbacGenerationException αποτελεί υποκλάση της κλάσης InternalServerErrorException και γίνεται throw όταν υπάρξει σφάλμα στην διαδικασία παραγωγής του XACML κώδικα για κάποιο ABAC Policy. Επιπλέον ο constructor της AbacGenerationException παίρνει σαν παράμετρο μια μεταβλητή τύπου String, η οποία περιέχει ένα μήνυμα σφάλματος το οποίο περιγράφει τους λόγους για τους οποίος δεν μπόρεσε να ολοκληρωθεί η διαδικασία παραγωγής του XACML κώδικα για κάποιο ABAC Policy. Ομοίως η κλάση XacmlRequestGenerationException αποτελεί υποκλάση της κλάσης InternalServerErrorException και γίνεται throw όταν υπάρξει σφάλμα κατά την διαδικασία παραγωγής του XACML Request. Ο constructor αυτής της κλάσης παίρνει σαν παράμετρο μια μεταβλητή τύπου String η οποία περιγράφει την αιτία του σφάλματος.
- **service**
  Το πακέτο service περιέχει κλάσεις οι οποίες υλοποιούν κάποιο Spring Service. Πιο συγκεκριμένα οι κλάσεις EnvironementService, ResourceService, ActionService και SubjectService, αποτελούν υποκλάσεις της κλάσης CategoryTemplate η οποία περιέχει μεθόδους για όλες τις βασικές λειτουργίες των κατηγοριών ABAC. Από την μεριά της η κλάση ABACService περιέχει μεθόδους για την διαχείριση των ABAC Policies όπως την δημιουργία, την επεξεργασία, την διαγραφή αλλά και την ανάγνωση των policies. Αξίζει να σημειωθεί ότι η συγκεκριμένη κλάση δεν περιέχει μεθόδους για την παραγωγή XACML κώδικα. Η κλάση ProjectService περιέχει μεθόδους για την διαχείριση του project όπως την αποθήκευση και την φόρτωσή του. Τέλος η κλάση XACMLService υλοποιεί μεθόδους για την παραγωγή ABAC Policy σε γλώσσα XACML αλλά και την παραγωγή XACML Request.

Τέλος η κλάση PolicyToolServiceApplication υλοποιεί την μέθοδο main η οποία αποτελεί αφετηρία εκτέλεσης του Back-End service.

3.3.3 Java 11

Η επιλογή της Java ως γλώσσα προγραμματισμού για την ανάπτυξη του Back-End Sevice δεν ήταν τυχαία. Έπειτα από ανάλυση των προδιαγραφών του χρήστη και βάσει των αναγκών που θα χρειαζόταν μία τέτοια πλατφόρμα η επιλογή μιας δυνατής αντικειμενοστραφούς γλώσσα προγραμματισμού ήταν απαραίτητη. Το γεγονός ότι υπήρχε η βιβλιοθήκη AuthzForce σε Java ήταν ο καθοριστικός λόγος για την επιλογή της γλώσσας.

Καθ' όλη την διαδικασία του development, χρησιμοποιήθηκαν patterns όπως dependency injection, DTO, inheritence, etc.

3.3.4 Spring Framework

Για την ανάπτυξη του Back-End Service χρησιμοποιήθηκε το Spring Framework [6][7][8][9]. Αξιοποιήθηκαν πλήρως οι δυνατότητες των Controllers και με αυτό τον τρόπο έγινε ευκολότερη η δημιουργία των endpoints.

Επιπλέον χρησιμοποιήθηκαν annotations για dependency injection καθώς και για τις ρυθμίσεις των Controllers και των Services. Η διαχείριση των σφαλμάτων από το API έγινε απλούστερη με την χρήση ControllerAdvice σε όλα τα endpoints. Όσο αφορά το input validation, δημιουργήθηκαν προσαρμοσμένες κλάσεις validation, όπου με την χρήση annotations εφαρμόστηκαν σε όλες τις παραμέτρους των endpoints όπου απαιτούσαν τέτοιου είδους επαλήθευση. Τέλος για την μεταφορά δεδομένων ανάμεσα στο Front-End και στο Back-End Serivce, αναπτύχθηκαν DTO κλάσεις, κάνοντας έτσι ευκολότερη την διαδικασία Serialization/Deserialization καθώς και την διαδικασία επαλήθευσης παραμέτρων των endpoints.

3.3.5 ReactJS

Για την σχεδίαση του Front-End Service χρησιμοποιήθηκε το ReactJS [10] σε συνδυασμό με την βιβλιοθήκη MUI [11] για την εύκολη δημιουργία React Components. Η γλώσσα προγραμματισμού που επιλέχθηκε ήταν η TypeScript [12] καθώς υποστηρίζει πολλές αντικειμενοστραφής μεθόδους προγραμματισμού, δίνοντας μας την δυνατότητα να εφαρμοστούν design patterns όμοια με αυτά που μπορούμε να εφαρμόσουμε σε γλώσσες όπως η Java. Επιπλέον λόγο του ότι η TypeScript είναι μια strong typed γλώσσα, έγινε ευκολότερος ο εντοπισμός και η διόρθωση συντακτικών σφαλμάτων. Τέλος για την διαχείριση των διαδρομών (routes) μέσα στην σελίδα χρησιμοποιήθηκε η βιβλιοθήκη React Route.

3.3.6 API

Το API specification το οποίο περιγράφει τα endpoints του Back-End Service, είναι γραμμένο σε μορφή YAML και ακολουθεί τις συμβάσεις του Swagger OpenAPI Specification 2.0 [13]. Όλα τα endpoitns έχουν κατηγοριοποιηθεί με την χρήση tags στις ακόλουθες κατηγορίες: Category, ABAC, XACML, Project.

- Το tag Category, περιέχει τα endpoints που αφορούν την διαχείριση των κατηγοριών Action, Environment, Resourcce και Subject.
- Το tag ABAC, περιέχει τα endpoints που αφορούν την διαχείριση των πολιτικών ABAC.
- Το tag XACML περιέχει τα endpoints που αφορούν την παραγωγή του XACML κώδικα.
- Το tag Project περιέχει τα endpoints που αφορούν την διαχείριση του project, όπως αποθήκευση και φόρτωση.

Τέλος έχουν οριστεί αντικείμενα για κάθε μορφής παραμέτρου που μπορεί να δεχθεί κάποιο API request.

Με αυτό τον τρόπο η παραγωγή client library σε οποιαδήποτε γλώσσα και framework γίνεται αρκετά απλή διαδικασία.

3.3.7 AuthzForce

Για την παραγωγή του XACML κώδικα, τόσο για XACML Requests όσο και για ABAC Policies χρησιμοποιήθηκε η βιβλιοθήκη AuthzForce [14]. Το project AuthzForce παρέχει πολλαπλά πακέτα και βιβλιοθήκες αλλά στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε το Java PAP API. Το συγκεκριμένο API παρέχει δυνατότητες δημιουργίας και διαχείρισης ABAC και XACML policies καθώς και όλες τις απαραίτητες κλάσεις και μεθόδους για την δημιουργία αυτών των αντικειμένων αλλά και την τελική παραγωγή του XACML κώδικα.

3.3.8 GSON

Προκειμένου να δίνεται η δυνατότητα αποθήκευσης και φόρτωσης του project, θα πρέπει να υπάρχει μία μέθοδος εξαγωγής όλων των δεδομένων που εισάγει ο χρήστης στην πλατφόρμα σε κάποια κωδικοποιημένη μορφή. Για την υλοποίηση αυτής της δυνατότητας χρησιμοποιήθηκε η βιβλιοθήκη GSON [15], όπου μας έδωσε την δυνατότητα να δημιουργούμε JSON αρχεία με τα δεδομένα που εισάγει ο χρήστης στην εφαρμογή. Επειδή όμως πολλές από τις κλάσεις που μας παρέχει το AuthzForce περιέχουν προσαρμοσμένους και περίπλοκους τύπους δεδομένων, με αποτέλεσμα η βιβλιοθήκη GSON να μην μπορεί να κάνει deserialize ορισμένες κλάσεις, έπρεπε να αναπτυχθούν ορισμένες βοηθητικές κλάσεις για το Serialization/Deserialization των δεδομένων αυτών.

3.3.9 Docker

Τόσο στο Back-End Service όσο και στο Front-End Service, υπάρχουν τα αντίστοιχα Dockerfile [17] και docker-compose.yaml [18] αρχεία για ευκολότερο deployment. Όλα τα στάδιο της ανάπτυξης της πλατφόρμας έχουν γίνει μέσα σε Docker [16] containers. Με αυτό το τρόπο αξιοποιήθηκαν όλα τα πλεονεκτήματα των containers και διασφαλίστηκε η λειτουργικότητα της εφαρμογής ανεξαρτήτως λειτουργικού συστήματος και άλλων εξωτερικών παραμέτρων. Επιπλέον είναι έγινε ευκολότερη η αυτοματοποιημένη διαδικασία του deployment μέσω του docker-compose αρχείου το οποίο έκτος από την δυνατότητα αυτόματου build των Dockerfiles, δημιουργεί και ορίζει εξωτερικές παραμέτρους της πλατφόρμας όπως networking, port forwarding, volumes, κτλ.

3.3.10 Back-End Service

Το Back-End service, είναι γραμμένο σε Java 11 και χρησιμοποιεί το Spring Framework προκειμένου να τρέξει το REST API. Όλα τα endpoints αποδέχονται μόνο HTTP requests και κάθε response ακολουθεί τις συμβάσεις του HTTP πρωτοκόλλου και το response body (όταν υπάρχει) είναι σε μορφή JSON. Σε περίπτωση κάποιου σφάλματος το response body έχει την ακόλουθη δομή:

Όνομα πεδίου	**Περιγραφή**
timestamp	Ημερομηνία και ώρα

status	Κωδικός κατάστασης HTTP
error	Τίτλος κωδικού κατάστασης HTTP
message	Μήνυμα σφάλματος

Για την δημιουργία των XACML Policies/Requests χρησιμοποιείται το AuthzForce Java SDK. Επιπλέον υπάρχει το API specification σε μορφή YAML το οποίο ακολουθεί τις συμβάσεις του Swagger. Το αρχείο βρίσκεται στο παρακάτω δημόσιο repository.[6]

3.3.11 Front-End Service

Το Front-End service, είναι γραμμένο σε TypeScript και τρέχει πάνω στο NodeJS [19]. Για το περιβάλλον διεπαφής χρήστη χρησιμοποιείται το ReactJS framework μαζί με το MUI Core. Προκειμένου να είναι εφικτή η επικοινωνία μεταξύ του Front-End service και του Back-End service, καλούνται οι απαραίτητες μέθοδοι και συναρτήσεις που έχουν παραχθεί από το Swagger OpenAPI Generator.

3.3.12 Επικοινωνία με το τοπικό σύστημα αρχείων

Βασικό χαρακτηριστικό της εφαρμογής, αποτελεί η δυνατότητα παραγωγής και αποθήκευσης των XACML Request/Policies. Για τον λόγο αυτό είναι απαραίτητη η επικοινωνία με το τοπικό σύστημα αρχείων. Αφού λοιπόν το Back-End service δημιουργήσει το XACML κώδικα, το Front-End service επικοινωνεί με το τοπικό σύστημα αρχείων και αποθηκεύει το XML αρχείο. Εκτός όμως από την αποθήκευση των XACML Requests/Policies, η επικοινωνία με το τοπικό σύστημα αρχείων είναι απαραίτητη και για την αποθήκευση ή τη φόρτωση του project. Για την αποθήκευση του project χρησιμοποιείται η βιβλιοθήκη GSON και ουσιαστικά το Back-End service παράγει και αποθηκεύει ένα αρχείο σε μορφή JSON, το οποίο εμπεριέχει όλες τις απαραίτητες πληροφορίες που τρέχουν εκείνη την στιγμή στο Back-End service. Για την φόρτωση του project το Back-End service διαβάζει το JSON αρχείο από το τοπικό σύστημα αρχείων και εκτελεί όλες τις απαραίτητες μεθόδους προκειμένου να φορτωθούν στο σύστημα όλες οι πληροφορίες.

6 https://github.com/gsoultos/policy-tool-service/blob/master/swagger.yaml

4. Παράδειγμα χρήσης

Για να γίνει ευκολότερα κατανοητή η χρήση της πλατφόρμας θα προσπαθήσουμε να εφαρμόσουμε τις λειτουργίες της σε ένα ρεαλιστικό παράδειγμα. Ας πάρουμε το παράδειγμα ενός νοσοκομείου. Θα έχουμε εργαζόμενους όπως γιατρούς, νοσοκόμους/νοσοκόμες, γραμματείς αλλά και τους ασθενείς. Όλοι οι παραπάνω θα πρέπει να έχουν πρόσβαση σε ορισμένες πληροφορίες και ανάλογα με τον ρόλο τους θα πρέπει να τους έχουν χορηγηθεί τα αντίστοιχα δικαιώματα. Δηλαδή θα υπάρχουν αρχεία όπως παλιό ιατρικό ιστορικό ασθενών, τωρινό ιστορικό ασθενών, συνταγογραφήσεις φαρμάκων, ραντεβού γιατρού - ασθενή και ιδιωτικές σημειώσεις γιατρών για τους ασθενείς τους. Για τις παραπάνω πληροφορίες θα πρέπει να οριστούν δικαιώματα πρόσβασης ανάλογα με τον ρόλο που έχει ο κάθε εργαζόμενος ή ασθενής στο νοσοκομείο. Επιπλέον θα πρέπει να οριστούν επιτρεπτές ενέργειες για τον κάθε ρόλο, για παράδειγμα μία νοσοκόμα θα μπορεί να δει το τωρινό ιατρικό ιστορικό ενός ασθενή αλλά δεν θα μπορεί να το τροποποιήσει. Τέλος η πρόσβαση στα δεδομένα θα πρέπει να γίνεται μέσω των υποδομών του νοσοκομείου, δηλαδή ένας νοσοκόμος θα μπορεί να έχει πρόσβαση στα στοιχεία των ασθενών μόνο όσο βρίσκεται εντός του νοσοκομείου, σε αντίθεση με τον ασθενή ο οποίος θα μπορεί να έχει πρόσβαση στα στοιχεία του από οπουδήποτε.

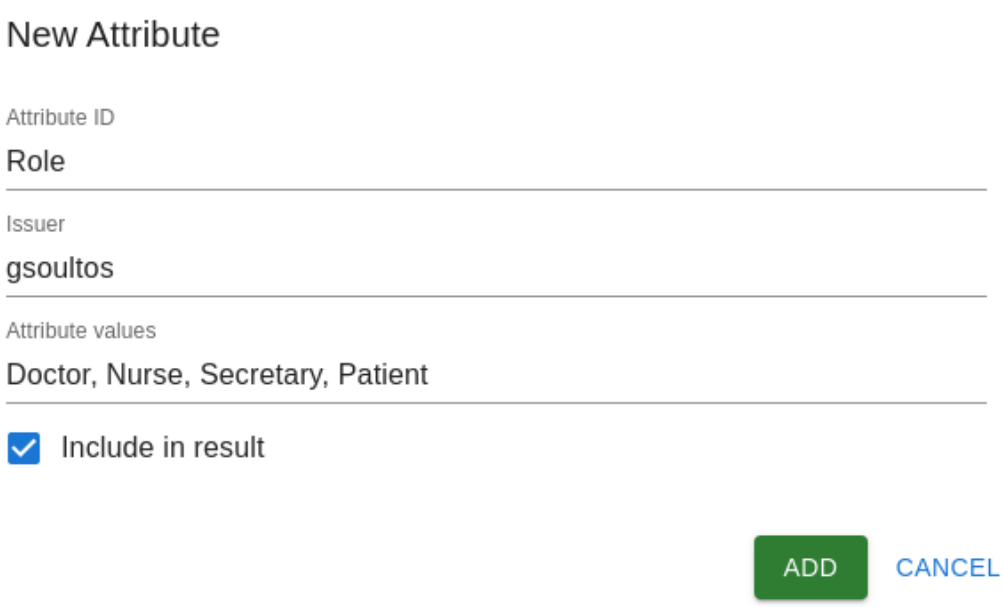
4.1 Δημιουργία των attributes και attribute values

Για κάθε κατηγορία (Subject, Resource, Action, Environment) δίνεται η δυνατότητα να δημιουργίας και να διαχείρισης των attributes και των attribute values.

Σχήμα 6: Βασικές κατηγορίες ABAC

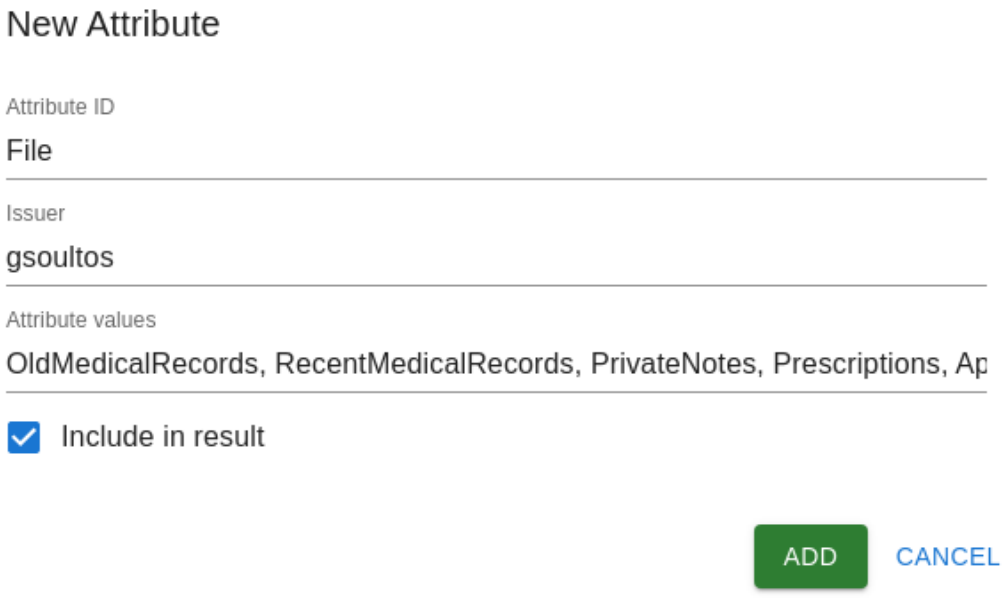
Έτσι ξεκινώντας θα πρέπει να δημιουργήσουμε τα απαραίτητα attributes και στην συνέχεια να προσθέσουμε τα attribute values.

Για την κατηγορία Subject θα δημιουργήσουμε ένα attribute με ID Role και θα προσθέσουμε σαν attribute values τους ρόλους που θα μπορεί να έχει οποιασδήποτε χρήστης του νοσοκομείου: Doctor, Nurse, Secretary, Patient.



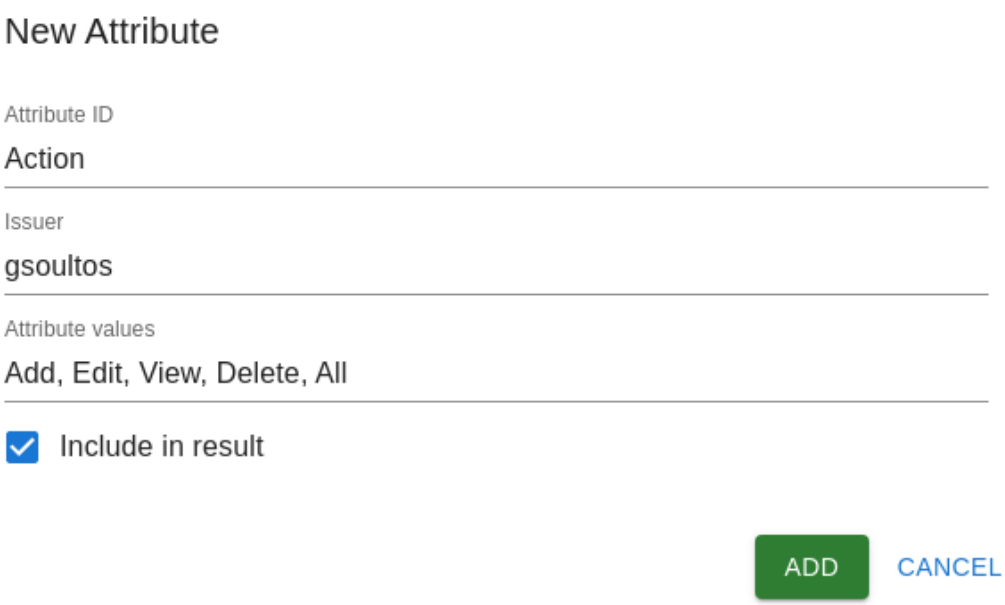
Σχήμα 7: Δημιουργία Role attribute

Για την κατηγορία Resource θα δημιουργήσουμε ένα attribute με ID File και θα προσθέσουμε σαν attribute values τις κατηγορίες των διαθέσιμων αρχείων που θα υπάρχουν στο νοσοκομείο: OldMedicalRecords, RecentMedicalRecords, PrivateNotes, Prescriptions, Appointment.



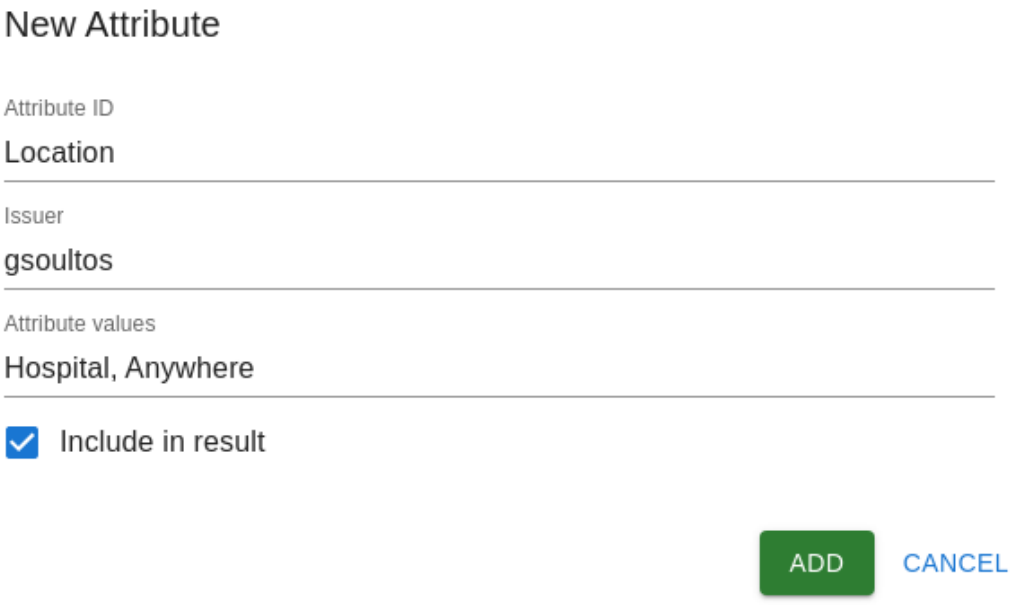
Σχήμα 8: Δημιουργία File attribute

Για την κατηγορία Action θα δημιουργήσουμε ένα attribute με ID Action και θα προσθέσουμε σαν attribute values τις επιλογές που θα μπορούν να έχουν οι χρήστες του νοσοκομείου πάνω στα αρχεία που ορίσαμε προηγούμενος στην κατηγορία Resource: Add, Edit, View, Delete, All.



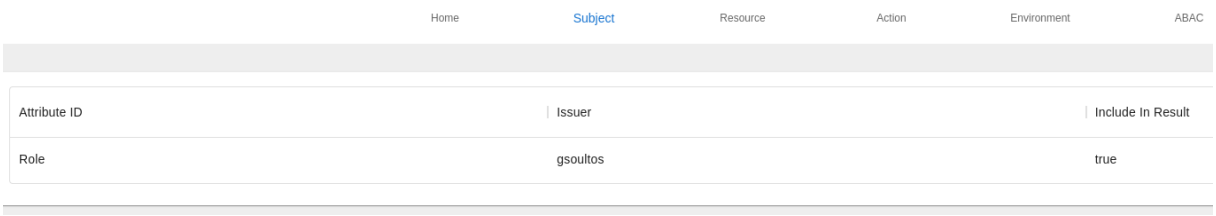
Σχήμα 9: Δημιουργία Action attribute

Τέλος για την κατηγορία Environment θα δημιουργήσουμε ένα attribute με ID Location και θα προσθέσουμε σαν attribute values τις διαθέσιμες τοποθεσίες από τις οποίες θα μπορούν οι χρήστες του νοσοκομείου να ζητήσουν πρόσβαση στα αρχεία του: Hospital, Anywhere.



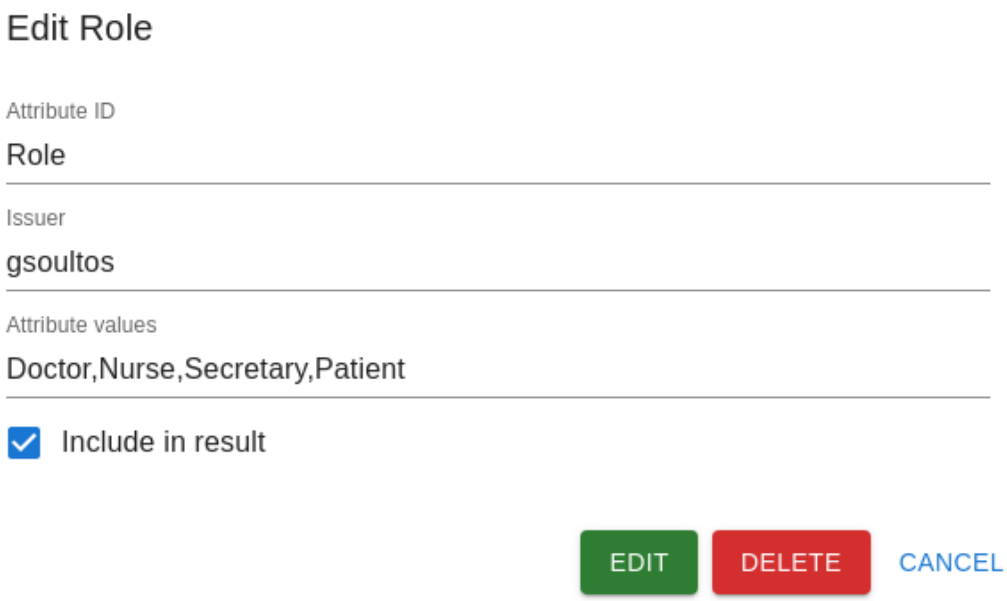
Σχήμα 10: Δημιουργία Location attribute

Στην συνέχεια μπορούμε να δούμε συγκεντρωμένα όλα τα υπάρχοντα attributes για την επιλεγμένη κατηγορία.



Σχήμα 11: Προβολή όλων των Subject attributes

Για την επεξεργασία ή την διαγραφή ενός attribute, αρκεί να κάνουμε αριστερό κλικ πάνω του, και στην συνέχεια μέσω ενός pop-up παραθύρου πανομοιότυπου με αυτού της δημιουργίας του, θα δίνεται η δυνατότητα τροποποίησης των τιμών του αλλά και της οριστικής διαγραφής του.

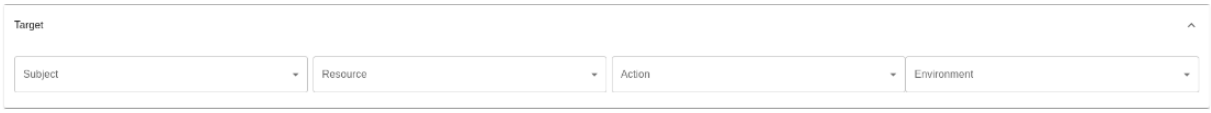


Σχήμα 12: Επεξεργασία του Role attribute

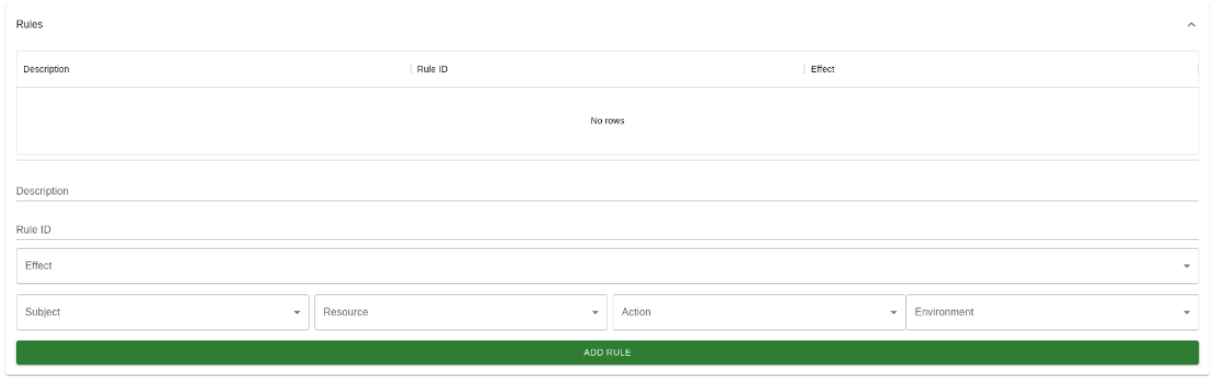
4.2 Διαχείριση ABAC

Επιλέγοντας από το μενού στο πάνω μέρος της οθόνης την επιλογή ABAC, θα εμφανίζεται η οθόνη δημιουργίας και διαχείρισης όλων των ABAC Policies. Έτσι πατώντας το κουμπί προσθήκης policy, θα εμφανίζεται η οθόνη δημιουργίας του ABAC Policy.

Σχήμα 13: Βασικές ρυθμίσεις πολιτικής



Σχήμα 14: Ρυθμίσεις Target για την πολιτική

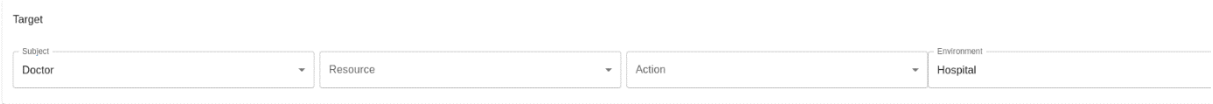


Σχήμα 15: Ρυθμίσεις Rules για την πολιτική

Στην συνέχεια για κάθε διαθέσιμο Role θα δημιουργήσουμε το αντίστοιχο policy. Έτσι ξεκινώντας με Doctor role θα ορίσουμε τις βασικές ρυθμίσεις του policy.

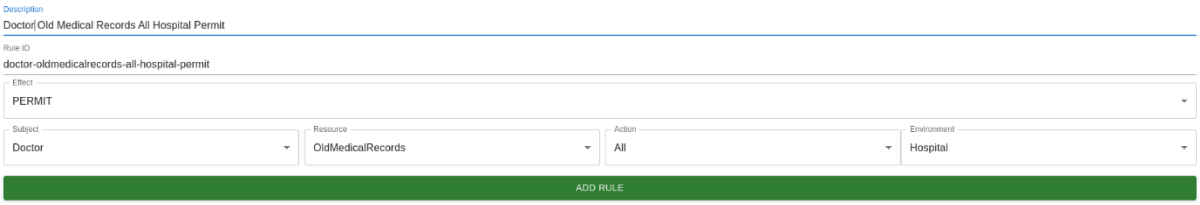
Σχήμα 16: Βασικές ρυθμίσεις για το Doctor policy

Ένας γιατρός θα μπορεί να έχει πρόσβαση προβολής σε όλα τα διαθέσιμα αρχεία που ορίσαμε προηγούμενος. Ωστόσο δεν θα μπορεί να τροποποιεί τα αρχεία ραντεβού των ασθενών αφού αυτά θα ορίζονται από την γραμματεία. Επιπλέον η πρόσβαση σε οποιοδήποτε αρχείο θα πρέπει να γίνεται μέσω των υποδομών του νοσοκομείου, έτσι θα πρέπει να υπάρχει περιορισμός και στην κατηγορία Environment. Συνεπώς το συγκεκριμένο policy θα εφαρμόζεται μόνο όταν το Target Subject έχει την τιμή Doctor και το Target Environment έχει την τιμή Hospital.

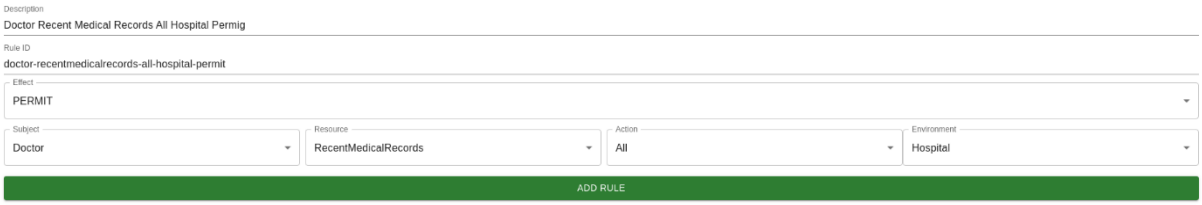


Σχήμα 17: Ρυθμίσεις Target για το doctor policy

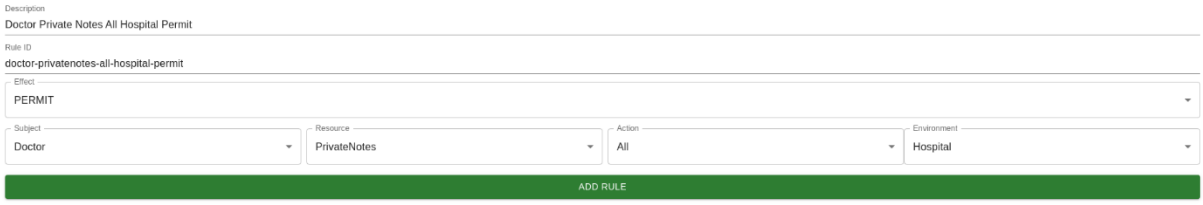
Τέλος, θα προσθέσουμε τα αντίστοιχα Rule policies όπως φαίνεται παρακάτω.



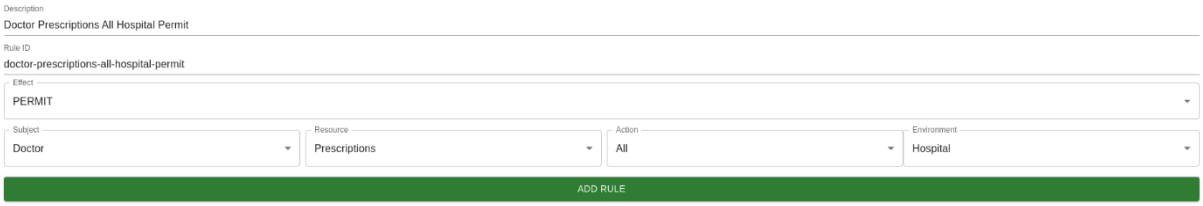
Σχήμα 18: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα παλιά ιατρικά δεδομένα όσο βρίσκεται εντός του νοσοκομείου



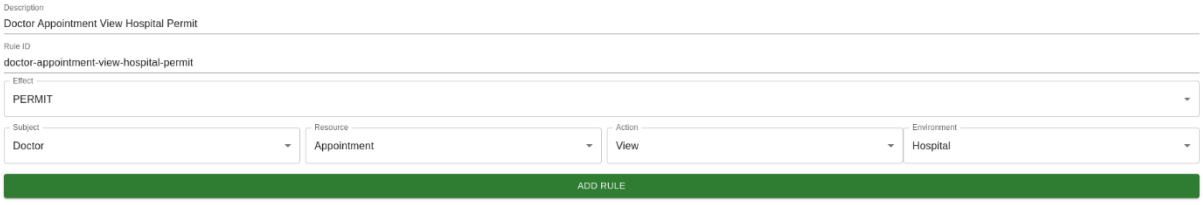
Σχήμα 19: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα πρόσφατα ιατρικά δεδομένα όσο βρίσκεται εντός του νοσοκομείου



Σχήμα 20: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στις ιδιωτικές του σημειώσεις όσο βρίσκεται εντός του νοσοκομείου

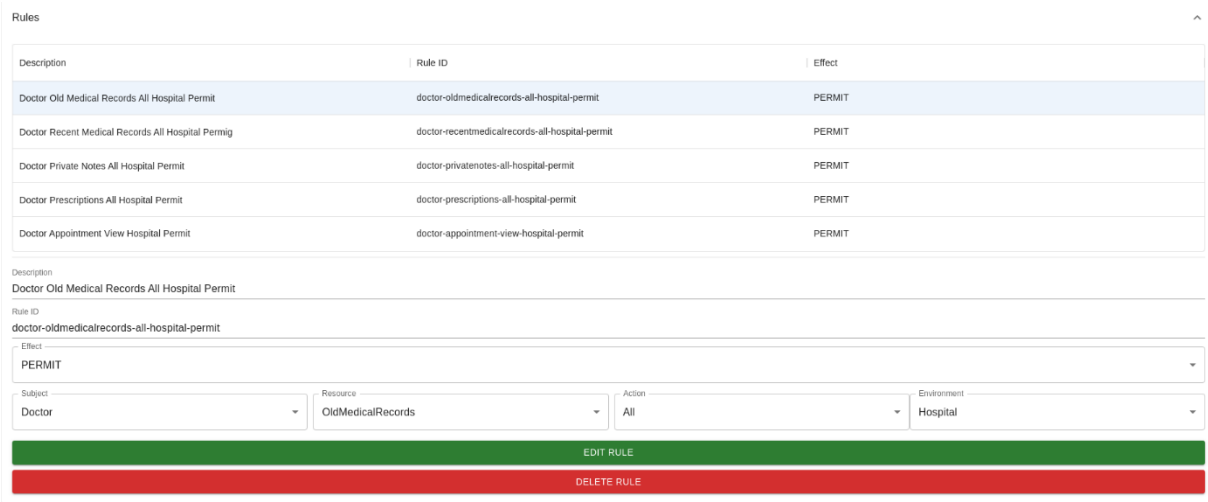


Σχήμα 21: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στις συνταγογραφίσεις όσο βρίσκεται εντός του νοσοκομείου



Σχήμα 22: Προσθήκη Rule που παραχωρεί στον γιατρό πλήρη πρόσβαση στα ραντεβού όσο βρίσκεται εντός του νοσοκομείου

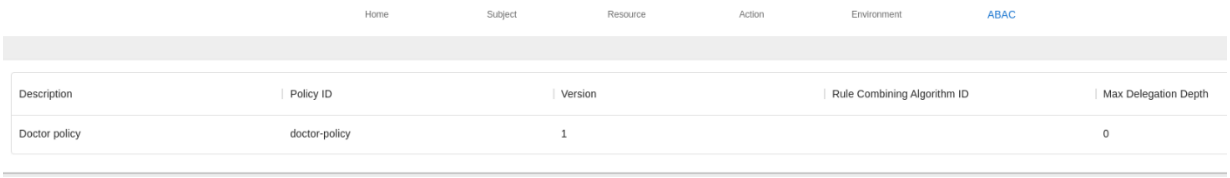
Μπορούμε να τροποποιήσουμε ή να διαγράψουμε οποιοδήποτε rule κάνοντας απλά αριστερό κλικ πάνω του και πατώντας το κουμπί επεξεργασίας ή διαγραφής αντίστοιχα.



Σχήμα 23: Επεξεργασία κανόνων του Policy

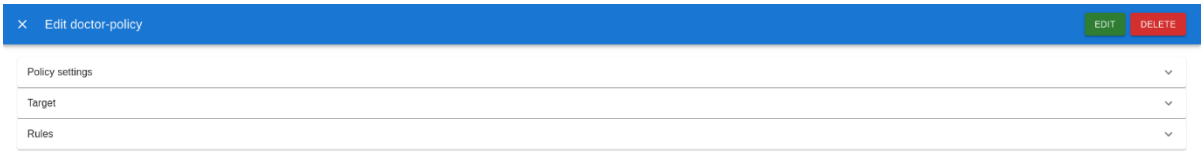
Αφού ολοκληρώσουμε την ρύθμιση του policy μπορούμε να πατήσουμε το κουμπί προσθήκης στο πάνω αριστερό μέρος της οθόνης για να δημιουργηθεί το καινούριο ABAC Policy.

Όλα τα ABAC Policies θα εμφανίζονται συγκεντρωμένα στην αρχική οθόνη ABAC.



Σχήμα 24: Προβολή όλων των ABAC policies

Κάνοντας κλικ πάνω σε κάποιο υπάρχον ABAC Policy, θα δίνεται η δυνατότητα επεξεργασίας ή διαγραφής του, μέσω μιας οθόνης όμοιας με αυτή της οθόνης δημιουργίας του policy.



Σχήμα 25: Επεξεργασία ABAC policy

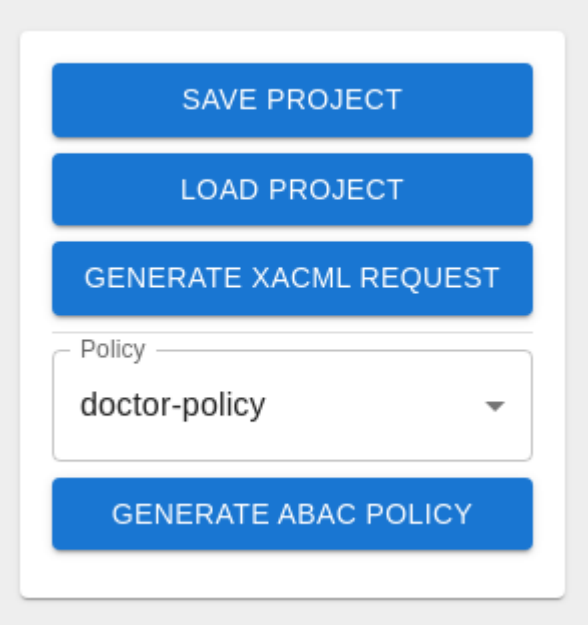
Ομοίως θα δημιουργήσουμε ABAC policies και για τους υπόλοιπους χρήστες του νοσοκομείου (Nurse, Secretary, Patient). Πιο συγκεκριμένα, μία νοσοκόμα θα μπορεί να έχει πρόσβαση προβολής στις ιατρικές συνταγές των ασθενών και μόνο όσο βρίσκεται εντός των υποδομών του νοσοκομείου. Από την άλλη μεριά μία γραμματέας θα πρέπει θα έχει πλήρη πρόσβαση μόνο στα δεδομένα των ραντεβού και μόνο όσο βρίσκεται εντός του νοσοκομείου. Τέλος ένας ασθενής θα πρέπει να έχει δικαίωμα προβολής σε όλα τα δεδομένα εκτός από τις ιδιωτικές σημειώσεις του γιατρού και φυσικά δεν θα υπάρχει περιορισμός στην τοποθεσία του.

4.3 Παραγωγή ABAC Request/Policy

Επιλέγοντας την αρχική οθόνη της εφαρμογής από το μενού στο πάνω μέρος της οθόνης, θα δίνεται η δυνατότητα παραγωγής XACML κώδικα για είτε για Request είτε για ABAC Policy.

Για την παραγωγή του XACML Request, αρκεί να έχουν δημιουργηθεί τα απαραίτητα attributes για τις διαθέσιμες κατηγορίες, και στην συνέχεια πατώντας το κουμπί "GENERARTE XACML REQUEST", θα παράγεται ο XACML κώδικας και θα δίνεται η δυνατότητα τοπικής αποθήκευσης σε μορφή XML αρχείου.

Ομοίως για την παραγωγή του ABAC Policy, επιλέγοντας ένα από τα υπάρχον ABAC Policies μέσω του drop-down menu, και πατώντας το κουμπί "GENERATE ABAC POLICY", θα παράγεται ο XACML κώδικας για το συγκεκριμένο ABAC Policy και θα δίνεται η δυνατότητα τοπικής αποθήκευσης σε μορφή XML αρχείου.



Σχήμα 26: Οθόνη παραγωγής ABAC Policy

Για το παραπάνω παράδειγμα του νοσοκομείου, ο XACML κώδικα που παράγεται για το doctor-policy είναι ο εξής:

<?xml version="1.0" encoding="UTF-8"?> <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="doctor-policy" Version="1" RuleCombiningAlgId="" MaxDelegationDepth="0"> <Description>Doctor policy</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" />

```
</Match> </AllOf> </AnyOf> </Target> <Rule RuleId="doctor-oldmedicalrecords-all-hospital-permit" Effect="Permit"> <Description>Doctor Old Medical Records All Hospital Permit</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">OldMedicalRecords</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="File" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">All</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="Action" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> </AllOf> </AnyOf> </Target> </Rule> <Rule RuleId="doctor-recentmedicalrecords-all-hospital-permit" Effect="Permit"> <Description>Doctor Recent Medical Records All Hospital Permig</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">RecentMedicalRecords</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="File" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">All</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="Action" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> </AllOf> </AnyOf> </Target> </Rule> <Rule RuleId="doctor-privatenotes-all-hospital-permit" Effect="Permit"> <Description>Doctor Private Notes All Hospital Permit</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue
```

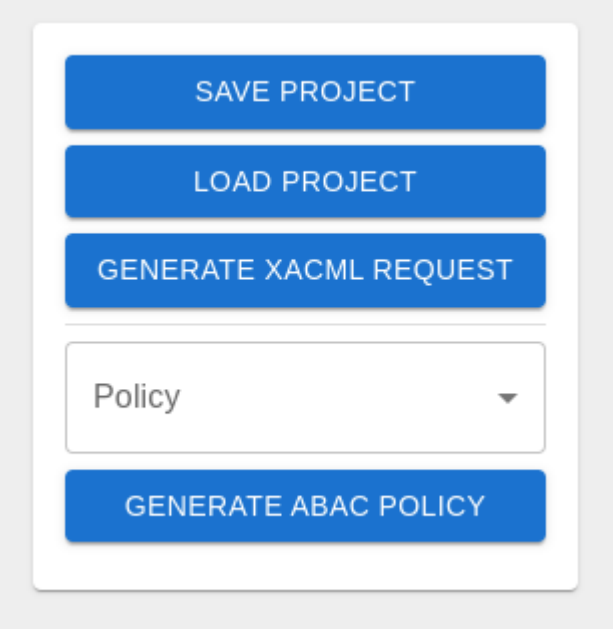
DataType="string">PrivateNotes</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="File" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">All</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="Action" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> </AllOf> </AnyOf> </Target> </Rule> <Rule RuleId="doctor-prescriptions-all-hospital-permit" Effect="Permit"> <Description>Doctor Prescriptions All Hospital Permit</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Prescriptions</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="File" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">All</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="Action" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> </AllOf> </AnyOf> </Target> </Rule> <Rule RuleId="doctor-appointment-view-hospital-permit" Effect="Permit"> <Description>Doctor Appointment View Hospital Permit</Description> <Target> <AnyOf> <AllOf> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Doctor</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:subject" AttributeId="Role" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">Appointment</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" AttributeId="File" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue DataType="string">View</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action" AttributeId="Action" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <AttributeValue

DataType="string">Hospital</AttributeValue> <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="Location" DataType="string" Issuer="gsoultos" MustBePresent="true" /> </Match> </AllOf> </AnyOf> </Target> </Rule> </Policy>

4.4 Διαχείριση project

Στην αρχική οθόνη της εφαρμογής, υπάρχει δυνατότητα αποθήκευσης και φόρτωσης project. Πατώντας το κουμπί "SAVE PROJECT" θα αποθηκεύονται όλα τα τρέχοντα δεδομένα της εφαρμογής τοπικά σε ένα αρχείο JSON.

Ομοίως για την φόρτωση κάποιου project, πατώντας το κουμπί "LOAD PROJECT" και επιλέγοντας το αποθηκευμένο JSON αρχείο η εφαρμογή φορτώνει όλα τα δεδομένα του project.



Σχήμα 27: Οθόνη διαχείρισης του project

5. Συμπεράσματα

Ο έλεγχος πρόσβασης αποτελεί σαφέστατα απαραίτητο στοιχείο της ασφάλειας και της ιδιωτικότητας των υπολογιστικών συστημάτων αλλά και των δεδομένων τους. Στο σύγχρονο πλαίσιο των μοντέρνων οργανισμών και επιχειρήσεων κρίνεται απαραίτητη η σωστή προσέγγιση των μέτρων ασφαλείας που απαιτούνται για τη διασφάλιση της ιδιωτικότητας των δεδομένων και των υποδομών. Ωστόσο όσο περιπλέκεται η εσωτερική τους δομή τόσο πιο δύσκολη γίνεται η ανάλυση και εφαρμογή πολιτικών ελέγχου πρόσβασης. Συνεπώς μέσω της πλατφόρμας που παρουσιάστηκε, δίνεται η δυνατότητα σε μη τεχνικούς χρήστες να δημιουργήσουν και να διαχειριστούν σωστά πολιτικές ελέγχου πρόσβασης βάσει χαρακτηριστικών που βασίζονται στις απαιτήσεις και στην εσωτερική δομή της εκάστοτε εταιρίας ή οργανισμού. Τέλος η αυτόματη παραγωγή του XACML κώδικα ελαχιστοποιεί τα περιθώρια λογικών σφαλμάτων κάνοντας ευκολότερη και απλούστερη την διαδικασία εφαρμογής των πολιτικών ελέγχου πρόσβασης.

6. Μελλοντική εργασία

Στην παρούσα πτυχιακή εργασία αναπτύχθηκε σύστημα διαχείρισης πολιτικών ελέγχου πρόσβασης βάσει χαρακτηριστικών. Η συγκεκριμένη πλατφόρμα αποτελεί μία απλοποιημένη μορφή PAP (Policy Administration Point) της αρχιτεκτονικής XACML. Η γλώσσα XACML αποτελεί ένα ολοκληρωμένο εργαλείο διαχείρισης πολιτικών ελέγχου πρόσβασης με πολλαπλές δυνατότητες και περιθώρια επέκτασης. Συνεπώς το συγκεκριμένο σύστημα επιδέχεται βελτιώσεις και δυνατότητες προσθήκης δυνατοτήτων τόσο στο κομμάτι της πλήρους αξιοποίησης της γλώσσας XACML όσο και στο περιβάλλον διεπαφής του χρήστη. Επιπλέον αξιόλογες προσθήκες δυνατοτήτων αποτελούν η διεπαφή με το σύστημα PDP (Policy Decision Point).

7. Βιβλιογραφία

1. S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, “Access Control: Principles and Solutions,” Softw. Pract. Exper., vol. 33, pp. 397–421, Apr. 2003.
2. S. De Capitani di Vimercati, P. Samarati, and R. Sandhu, “Access Control,” in Computer Science Handbook (3rd edition) - Information Systems and Information Technology (A. Tucker and H. Topi, eds.), Taylor and Francis Group, 2014. to appear.
3. D.F. Ferraiolo and R. Sandhu. Proposed nist standard for role-based access control.ACM Transactions on Information and System Security (TISSEC), 4(3):224–274, August 2001.
4. E. Yuan and J. Tong, “AÎributed Based Access Control (ABAC) for Web Services,” in ICWS ’05: Proceedings of the IEEE International Conference on Web Services, 2005.
5. Organization for the Advancement of Structured Information Standards (OASIS), “eXtensible Access Control Markup Language (XACML) Version 3.0”, OASIS Standard, January 2013, https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
6. Spring Framework “Core Technologies” 5.3.23 https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring-core
7. Spring Framework “Testing” 5.3.23 https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html#testing
8. Spring Framework “Data Access” 5.3.23 https://docs.spring.io/spring-framework/docs/current/reference/html/data-access.html#spring-data-tier
9. Spring Framework “Web on Servlet Stack” https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#spring-web
10. ReactJS https://reactjs.org/docs/getting-started.html
11. MUI “Core” 5.10.9 https://mui.com/material-ui/getting-started/overview/
12. TypeScript https://www.typescriptlang.org/docs/
13. Swagger “OpenAPI Specification” 2.0 https://swagger.io/specification/v2/
14. AuthzForce 7.1.0 https://www.javadoc.io/doc/org.ow2.authzforce/authzforce-ce-core/7.1.0/index.html
15. GSON 2.8.9 https://www.javadoc.io/doc/com.google.code.gson/gson/2.8.9/com.google.gson/module-summary.html
16. Docker https://docs.docker.com/reference/
17. Dockerfile https://docs.docker.com/engine/reference/builder/
18. Docker Compose https://docs.docker.com/compose/compose-file/
19. NodeJS https://nodejs.org/docs/latest-v16.x/api/