ΕΚΤΙΜΗΣΗ ΑΝΤΙΚΤΥΠΟΥ ΣΤΗΝ ΙΔΙΩΤΙΚΟΤΗΤΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΤΑΝΕΜΗΜΕΝΟΥ ΚΑΘΟΛΙΚΟΥ

ΤΕΡΕΖΑΚΗ ΑΡΙΣΤΕΑ

B.S., ΑΤΕΙ Πατρών, 2005

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΕΠΙΣΤΗΜΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΤΕΠΙΣΤΗΜΙΟ

2022

Εγκρίθηκε από:

Υπεύθυνος Καθηγητής:
Ευάγγελος Μαρκάκης

Abstract

The technological progress has advanced the quantity of personal data organizations and companies can globally acquire. A situation that brought to light the inability of Directive 95/46/EC, the legal act concerned the personal data processing in the European Union (EU) since 1995, incited the European Commission to adopt, in April of 2016, the General Data Protection Regulation (GDPR). The enforcement of the Regulation started in May of 2018, introducing new obligations to the parties that process personal data and new rights to the European citizens whose personal information are being processed. Along with GDPR, the Data Protection Impact Assessment (DPIA) reintroduced to the corporate world. A process that was known worldwide and GDPR enhanced to a protection and controlling mechanism for the products and projects involved in personal data processing. However, GDPR does not offer particular information regarding the form of a DPIA. Because of the DPIA's preliminary existence, there are several forms and applications available.

Undoubtedly, the protection of personal data is an immense subject that concerns everyone nowadays. Howbeit, there is a matter that has not received the appropriate attention. Specifically, how the authenticity of a DPIA outcome can be assured. That is the question this project attempts to answer. The accomplishment of the task was achieved by developing a new application that automatically stores the DPIA outcome in a private blockchain. The project is structured based on the application CNIL PIA of the French Commission Nationale de l'Informatique et des Liberte which is used to carry out the DPIA, and the blockchain technology which provides immutability to all the records that it contains.

Περίληψη

Η τεχνολογική εξέλιξη των τελευταίων δεκαετιών πυροδότησε την συλλογή προσωπικών δεδομένων, γεγονός που ανάδειξε την αδυναμία της ισχύουσας νομοθεσίας περί προστασίας των προσωπικών δεδομένων και ώθησε την ευρωπαϊκή επιτροπή να προβεί στην σύνταξη του Γενικού Κανονισμού Προσωπικών Δεδομένων (ΓΚΠΔ). Το 2018 ξεκίνησε να εφαρμόζεται, επιφέροντας, με τον υποχρεωτικό του χαρακτήρα, τον επαναπροσδιορισμό της αγοράς ως προς την διαχείριση των δεδομένων προσωπικού χαρακτήρα. Με τον ΓΚΠΔ, μεταξύ άλλων, συστήθηκε στο ευρύτερο επιχειρηματικό σύνολο η Εκτίμηση Αντικτύπου Προστασίας Δεδομένων (ΕΑΠΔ). Μια διαδικασία που προϋπήρχε αλλά δεν αποτελούσε βασικό μηχανισμό πρόληψης και ελέγχου των προϊόντων και των υπηρεσιών που σχετίζονται με την χρήση ευαίσθητων προσωπικών δεδομένων. Ενώ ο ΓΚΠΔ δεν έχει ορίσει κάποιο συγκεκριμένο πρότυπο ή εργαλείο σύνταξης και υλοποίησης της ΕΑΠΔ, υπάρχουν αρκετά πρότυπα και εργαλεία που μπορούν να χρησιμοποιηθούν. Ωστόσο, ενώ η σημαντικότητα διαφύλαξης των προσωπικών δεδομένων έχει έρθει στο προσκήνιο και αποτελεί ίσως ένα από τα πιο φλέγοντα ζητήματα της εποχής, υπάρχει ένα θέμα που δεν φαίνεται να έχει λάβει την αντίστοιχη προσοχή. Η διασφάλιση της αυθεντικότητας μιας ΕΑΠΔ. Πάνω σε αυτό το πρόβλημα δομήθηκε η παρούσα εργασία. Επιλέχθηκε ένα από τα υπάρχοντα εργαλεία, το CNIL PIA για την δημιουργία μιας Εκτίμησης Αντικτύπου. Μια εφαρμογή του French Commission Nationale de l'Informatique et des Liberte, η οποία έχει κατασκευαστεί για να παράγει μια ολοκληρωμένη Εκτίμηση Αντικτύπου. Χρησιμοποιώντας την CNIL PIA και την τεχνολογία blockchain που σαν βασικό χαρακτηριστικό της έχει την αμεταβλητότητα των δεδομένων που καταχωρούνται σ' αυτό, δημιουργήθηκε μια εφαρμογή που αποθηκεύει αυτόματα το αποτέλεσμα της CNIL PIA καθώς και των ενεργειών που προκύπτουν σε ένα ιδιωτικό blockchain.

Πίνακας περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

Κεφάλαιο 1- Εισαγωγή

Γενικός Κανονισμός για την Προστασία Δεδομένων (ΓΚΠΔ)

Η ευρωπαϊκή επιτροπή, μέσα από σκληρές διαπραγματεύσεις μεταξύ των μελών της και παρατηρώντας τις νέες συνθήκες που αφορούν στα προσωπικά δεδομένα των φυσικών προσώπων, ωθήθηκε να νομοθετήσει και να εφαρμόσει λίγο αργότερα, το Γενικό Κανονισμό για την Προστασία Δεδομένων (General Data Protection Regulation- GDPR). Πρόκειται για έναν κανονισμό που πραγματεύεται την προστασία δεδομένων προσωπικού χαρακτήρα και τις ελευθερίες των Ευρωπαίων πολιτών, χωρίς να θίγει τον τόπο κατοικίας τους. Ο κανονισμός εφαρμόστηκε τον Μάιο του 2018 και ουσιαστικά αντικατέστησε την νομοθεσία που ήταν σε ισχύ από το 1995, την Οδηγία 95/46/ΕΚ.

Ο Κανονισμός αποτελεί ένα κοινό πλαίσιο ρυθμίσεων για τον τρόπο με τον οποίο συλλέγονται, επεξεργάζονται, φυλάσσονται, διακινούνται, αξιοποιούνται αλλά και καταστρέφονται δεδομένα προσωπικού χαρακτήρα, τόσο σε ηλεκτρονική όσο και σε φυσική μορφή. [1] Ουσιαστικά σαν στόχο του έχει να βρεθεί η χρυσή τομή ανάμεσα στην προστασία της ιδιωτικότητας των φυσικών προσώπων και την ελεύθερη διακίνηση των δεδομένων αυτών [2].

Οι παράμετροι που οδήγησαν στον νέο Κανονισμό, ήταν κατά κύριο λόγο, οι ραγδαίες τεχνολογικές εξελίξεις και η ασυμμετρία εφαρμογής της Οδηγίας 95/46/ΕΚ από τα κράτη μέλη. Παρατηρήθηκε μεγάλη αύξηση της συλλογής, ανταλλαγής και επεξεργασίας των προσωπικών δεδομένων καθώς και αύξηση των περιπτώσεων όπου παραβιάστηκε η ασφάλειας των δεδομένων αυτών. Ενώ την ίδια στιγμή, είχαν αρχίσει να διαφαίνονται οι αποκλίσεις από την εκτέλεση και την εφαρμογή του μέχρι τότε νόμου. [1]

Βασικές προσθήκες του ΓΚΠΔ

- Η συγκατάθεση του υποκειμένου για την επεξεργασία των προσωπικών του δεδομένων.
- Η διαφάνεια στην επεξεργασία και διευκρίνιση της αρχής της ελαχιστοποίησης των δεδομένων.
- Αυστηροί κανόνες όταν πρόκειται για επεξεργασία δεδομένων παιδιών.
- Το δικαιώματος διαγραφής και το δικαιώματος φορητότητας των δεδομένων.

- Το δικαίωμα αντίρρησης στη δημιουργία προφίλ ενός υποκειμένου και την αυτοματοποιημένη λήψη αποφάσεων.
- Ο υπεύθυνος επεξεργασίας δεδομένων.
- Η υποχρέωση των υπευθύνων να διενεργούν εκτίμηση των επιπτώσεων σχετικά με την προστασία των δεδομένων. [3]

Δεδομένα Προσωπικού Χαρακτήρα

Σύμφωνα με το Άρθρο 4 του Γενικού Κανονισμού, ως προσωπικά δεδομένα ορίζονται οι πληροφορίες εκείνες που μπορούν έμμεσα ή άμεσα να ταυτοποιήσουν ένα φυσικό πρόσωπο («υποκείμενο των δεδομένων») μέσω κάποιου αναγνωριστικού στοιχείου ταυτότητας, δεδομένων θέσης, επιγραμματικού αναγνωριστικού ταυτότητας ή παραγόντων που προσδιορίζουν την σωματική, φυσιολογική, γενετική, ψυχολογική, οικονομική, πολιτιστική ή κοινωνική ταυτότητα του αναφερόμενου φυσικού προσώπου. Χαρακτηριστικά παραδείγματα αποτελούν το όνομα, το επώνυμο, ο αριθμός ταυτότητας, ΑΜΚΑ, ΑΦΜ, τηλέφωνο κτλ. [4]

Επεξεργασία Δεδομένων

Η επεξεργασία δεδομένων, βάση του άρθρου 4 (2) και (6) του Γενικού Κανονισμού Προσωπικών Δεδομένων, αναφέρεται και αφορά ένα σύνολο διαδικασιών όπως η συλλογή, η καταγραφή, η οργάνωση, η συγκρότηση, η αποθήκευση, η υιοθέτηση ή αλλαγή, η ανάκτηση, η γνωμοδότηση, η χρήση, η δημοσίευση κατά τη μεταφορά, η διάδοση και η διαθεσιμότητα, ο περιορισμός, η διαγραφή ή καταστροφή των προσωπικών δεδομένων.

Παραδείγματα επεξεργασίας μπορούν να θεωρηθούν, ο διαμοιρασμός εγγράφων που περιλαμβάνονται προσωπικά δεδομένα, η ανάρτηση φωτογραφικού υλικού ενός φυσικού προσώπου στο διαδίκτυο, η καταγραφή βίντεο κτλ. [5]

Εκτίμηση Αντικτύπου σχετικά με την Προστασία Δεδομένων (ΕΑΠΔ)

Ο Γενικός Κανονισμός, μεταξύ άλλων, παρουσιάζει την Εκτίμηση Αντικτύπου Προσωπικών Δεδομένων (Data Protection Impact Assessment- DPIA). Πρόκειται για ένα μηχανισμό πρόληψης, που λειτουργεί ως παράγοντας στάθμισης και συμμόρφωσης του εκάστοτε σχεδίου ή προϊόντος, σύμφωνα με τις επιταγές του Κανονισμού. Η ΕΑΠΔ ουσιαστικά καθορίζει με το αποτέλεσμα της, αν το προϊόν είναι συμβατό με όσα έχει ορίσει ο Κανονισμός ως προς την προστασία των δικαιωμάτων και των ελευθερίων των φυσικών προσώπων.

Η Εκτίμηση Αντικτύπου δεν σχεδιάστηκε για τον Κανονισμό, προϋπήρχε από τα τέλη του 1980 με μικρές διαφοροποιήσεις. Γίνεται αναφορά λοιπόν, για μια διαδικασία που με βάση τα άρθρα 35 και 36 του Κανονισμού, δεν έχει συγκεκριμένο πλαίσιο, μεθοδολογία ή πρότυπο, βάση του οποίου θα πρέπει να περατωθεί. Τίθενται ωστόσο κάποιες περιπτώσεις που η εφαρμογή της προτείνεται να εκτελείται. [6]

Όπως προαναφέρθηκε η Εκτίμηση Αντικτύπου είναι μια βοηθητική διαδικασία. Έχει σχεδιαστεί για να αναλύει, να διακρίνει και κατά συνέπεια να συμβάλλει στην ελαχιστοποίηση των ενδεχομένων κινδύνων που θα μπορούσαν να προκύψουν μέσα από την επεξεργασία των προσωπικών δεδομένων. Σε καμία περίπτωση όμως, δεν έχει την δυνατότητα να εξαλείψει όλους τους πιθανούς κινδύνους. Έχει ανιχνευτικό χαρακτήρα. Παράλληλα, δεδομένης της πολυπλοκότητας του σχεδιασμού και της υλοποίησης ενός προϊόντος, με την Εκτίμηση Αντικτύπου υποδεικνύει και η ανάληψη των ευθυνών κατά περίπτωση, μέσα στα πλαίσια που έχει καθορίσει ο Κανονισμός. Η αδυναμία εκπλήρωσης μιας Εκτίμησης Αντικτύπου, μπορεί να έχει ως συνέπεια την επιβολή προστίμου το οποίο μπορεί να ανέλθει στα 20 εκατομμύρια ευρώ ή όταν πρόκειται για επιχείρηση, στο 4% του συνολικού παγκόσμιου ετήσιου κύκλου εργασιών του προηγούμενου οικονομικού έτους.

Πέραν της εξομάλυνσης των πιθανών κινδύνων και την συμμόρφωση με τον Γενικό Κανονισμό, υπάρχουν πρόσθετα θετικά στοιχεία που μπορεί να συμβάλλει η Εκτίμηση Αντικτύπου. Η βελτίωση της ποιότητας των προϊόντων για παράδειγμα και κατά συνέπεια σε ένα πιο γενικότερο πλαίσιο, η ποιοτική εξέλιξη ολόκληρης της αγοράς. Δεδομένου του συμβουλευτικό της χαρακτήρα προσφέρει διαφάνεια που με την σειρά της συμβάλλει στην δημιουργία σχέσεων βασιζόμενων στην ασφάλεια και την διάθεση αφοσίωσης ανάμεσα σε επιχειρήσεις και πελάτες. Οι πελάτες πλέον καταλαβαίνουν πως και γιατί χρησιμοποιούνται τα προσωπικά τους δεδομένα, καθώς και οι επιχειρήσεις, είναι σε θέση μετά από αυτό ν' αντιληφθούν τις πραγματικές ανάγκες των πελατών τους, τις προσδοκίες και τις ανησυχίες τους.[7]

Διεξαγωγή μιας Εκτίμησης Αντικτύπου

Γνωρίζοντας τον προστατευτικό χαρακτήρα της Εκτίμησης Αντικτύπου, κρίνεται ωφέλιμο να πραγματοποιείται σε αρχικό στάδιο υλοποίησης ενός προϊόντος, παράλληλα με τον σχεδιασμό και την ανάπτυξη του. Δεν έχει στατική υπόσταση. Έχει δυναμικό χαρακτήρα και

προκειμένου να διατηρήσει και να διασφαλίσει την ακεραιότητα της, κρίνεται αναγκαία η ενημέρωση και η επανάληψη της όταν προκύψουν αλλαγές που επηρεάζουν το αποτέλεσμα του προϊόντος για το οποίο έχει εφαρμοστεί. Υπάρχουν κάποια ενδεικτικά βήματα που θα πρέπει ο υπεύθυνος επεξεργασίας να ακολουθήσει για την ολοκλήρωση μιας Εκτίμησης Αντικτύπου. Ωστόσο, ενδέχεται να υπάρξουν κάποιες διαφοροποιήσεις ανάλογα με το εργαλείο που επιλέγεται να χρησιμοποιηθεί.

Βήμα 1ο: Αναγνώριση της ανάγκης πραγματοποίησης μιας Εκτίμησης Αντικτύπου. Έχουμε αναφέρει ότι η Εκτίμηση Αντικτύπου δεν έχει υποχρεωτικό χαρακτήρα παρά μόνο σε πολύ συγκεκριμένες περιπτώσεις, όπως:

- σε μια συστηματική και εκτενή εκτίμηση των προσωπικών πτυχών φυσικού προσώπου, συμπεριλαμβανομένης της κατάρτισης προφίλ
- στην επεξεργασία ευαίσθητων δεδομένων σε μεγάλη κλίμακα
- στην συστηματική παρακολούθηση δημοσίων χώρων σε μεγάλη κλίμακα[8]

Όταν κάποιος δεν γνωρίζει αν το προϊόν του ανήκει στις προαναφερθείσες περιπτώσεις, θα πρέπει εφόσον η επιχείρηση ή ο φορέας διαθέτει Υπεύθυνο Προστασίας Δεδομένων (Data Protection Officer- DPO) να τον συμβουλευτεί. Διαφορετικά, να ελέγξει αν η επεξεργασία των δεδομένων που πρόκειται να κάνει, ενδέχεται να καταλήξει σε υψηλό κίνδυνο. Καλή πολιτική είθισται να θεωρείται η πραγματοποίηση της Εκτίμησης Αντικτύπου στις περιπτώσεις που ο υπεύθυνος διατηρεί αμφιβολίες.

Βήμα 2ο: Περιγραφή της επεξεργασίας.

Θα πρέπει ο υπεύθυνος σε αυτό το σημείο να περιγράψει τον τρόπο που θα γίνει η επεξεργασία των προσωπικών δεδομένων αλλά και τον σκοπό που πρόκειται να χρησιμοποιηθούν τα δεδομένα προσωπικού χαρακτήρα. Η περιγραφή θα πρέπει να αναφέρει, την μορφή της επεξεργασίας, την έκταση, το περιεχόμενο αλλά και τον σκοπό της.

Βήμα 3ο: Συμπερίληψη της άποψης των εμπλεκομένων στην διαδικασία προσώπων.

Στις περιπτώσεις εκείνες που η επεξεργασία των προσωπικών δεδομένων αφορά υπαρκτές επαφές, είναι φρόνιμο να σχεδιαστεί μια διαδικασία με συμβουλευτικό χαρακτήρα, που θα συλλέγονται οι απόψεις των συγκεκριμένων προσώπων ή εκπροσώπων αυτών.

Βήμα 4ο: Αξιολόγηση αναγκαιότητας και αναλογικότητας.

Ο υπεύθυνος θα πρέπει να αναλογιστεί σε αυτό το σημείο αν το σχέδιο που έχει αναπτύξει, είναι το κατάλληλο για να επιτευχθεί ο στόχος ή αν υπάρχει εναλλακτικός τρόπος που θα μπορούσε να έχει το ίδιο αποτέλεσμα.

Βήμα 5ο: Αναγνώριση και αξιολόγηση των κινδύνων.

Εδώ θα πρέπει να αναγνωριστούν οι ενδεχόμενοι κίνδυνοι που θα είχαν αρνητικό αντίκτυπο, σωματικό, συναισθηματικό ή υλικό, στα φυσικά πρόσωπα που ανήκουν τα προσωπικά δεδομένα τα οποία θα υποστούν επεξεργασία.

Βήμα 6ο: Ανίχνευση μέτρων πρόληψης από τους ενδεχόμενους κινδύνους.

Την ανίχνευση και τον εντοπισμό των κινδύνων που θα μπορούσαν να επιφέρουν αρνητικές συνέπειες στην ζωή των προσώπων που ανήκουν τα δεδομένα προσωπικού χαρακτήρα. Σε αυτό το βήμα, ακολουθεί η διαδικασία όπου για κάθε έναν από τους κινδύνους αυτούς θα πρέπει ο υπεύθυνος να εντοπίσει τις επιλογές που προληπτικά θα εφαρμόσει ώστε να εξαλείψει ή μειώσει τον κίνδυνο.

Βήμα 7ο: Επικύρωση και καταγραφή αποτελέσματος.

Σε αυτό το σημείο, θα πρέπει να γίνει καταγραφή όλων των πρόσθετων μέτρων που πρόκειται να ληφθούν και να εφαρμοστούν για την προστασία των προσωπικών δεδομένων καθώς επίσης και να καταγραφούν αναλυτικά, οι κίνδυνοι που έχουν εντοπιστεί, αποκλείστηκαν, αν μειώθηκε η πιθανότητα να συμβούν ή αν τελικά έγινε αποδοχή κάποιου εξ αυτών, είτε γιατί δεν είναι εφικτός ο περιορισμός τους, είτε γιατί το κόστος του περιορισμού είναι πολύ υψηλό. Θα πρέπει επίσης να συμβουλευτεί ο υπεύθυνος επεξεργασίας τον υπεύθυνο προστασίας δεδομένων, εφόσον υπάρχει και να καταγράψει την άποψη του ακόμα κι αν αποφασίσει να μην την λάβει υπόψη του. Σε αυτή την περίπτωση θα πρέπει να καταγράψει τους λόγους που επέλεξε να παραβλέψει τις συμβουλές του. [9] Γενικότερα, φρόνιμο είναι ο υπεύθυνος επεξεργασίας να είναι ιδιαίτερα αναλυτικός καταγράφοντας κάθε λεπτομέρεια για τις επιλογές που ακολούθησε αλλά και τους λόγους που επέλεξε να το κάνει.

Υπεύθυνο Προστασίας Δεδομένων (Data Protection Officer- DPO)

Ο Υπεύθυνος Προστασίας Δεδομένων, έχει σαν βασικό ρόλο την παρακολούθηση συμμόρφωσης του οργανισμού ή της επιχείρησης με τις επιταγές του Γενικού Κανονισμού Προσωπικών Δεδομένων. Ο ρόλος του είναι συμβουλευτικός και σαν κύριο μέλημα του έχει την

ενημέρωση του υπεύθυνου επεξεργασίας για τις υποχρεώσεις του αλλά και να παρέχει συμβουλές αναφορικά με την Εκτίμηση Αντικτύπου όταν του ζητηθεί.

Ο ορισμός Υπεύθυνου Προστασίας δεν είναι υποχρεωτικός παρά μόνο σε συγκεκριμένες περιπτώσεις, όπως όταν η επεξεργασία δεδομένων γίνεται από δημόσιο φορέα ή δημόσια αρχή, ή όταν πρόκειται για πράξεις επεξεργασίας που απαιτούν τακτική και συστηματική παρακολούθηση των υποκειμένων των προσωπικών δεδομένων σε μεγάλη κλίμακα ή επίσης όταν αφορά την επεξεργασία σε μεγάλη κλίμακα ειδικών κατηγοριών δεδομένων προσωπικού χαρακτήρα.[10]

Υψηλός Κίνδυνος (High Risk)

Αναφορικά με την επεξεργασία των προσωπικών δεδομένων, σύμφωνα με την Γενικό Κανονισμό, ως κίνδυνος ορίζεται ένα σενάριο που περιγράφει ένα γεγονός και τις συνέπειες του, εκτιμώμενο βάση της σοβαρότητας και της πιθανότητας. [11] Ο κίνδυνος ως προς τα δικαιώματα και τις ελευθερίες των φυσικών προσώπων μπορεί να οδηγήσει σε σωματικές, υλικές και άυλες επιπτώσεις. [12]

Εργαλεία και πρότυπα DPIA

Η Εκτίμησης Αντικτύπου έχει ωθήσει αρκετούς οργανισμούς να σχεδιάσουν και να δημιουργήσουν πρότυπα και εφαρμογές που θα μπορούσαν να απλοποιήσουν την εκτέλεση της. Μερικά εκ των οποίων είναι: UK PIA code of practice (UK Information Commissioner’s Office (ICO)) [13], New Zealand PIA toolkit (Office of Privacy Commissioner of New Zealand) [14], Australian ICO PIA guide (Office of the Australian Information Commissioner (OAIC)) [15], CNIL PIA (French Commission Nationale de l’Informatique et des Libertes) [16], Canada directive on PIA (Treasury Board of Canada Secretariat (Canada TBS)) [17] και ISO 29134 (International Organization for Standardization (ISO)) [18]

Κεφάλαιο 2- Τεχνολογία Κατανεμημένου Καθολικού (Distributed Ledger Technology)

Μέχρι και σήμερα, η αποθήκευση, η οργάνωση και η ως ενός βαθμού επεξεργασία των δεδομένων, γίνεται με την χρήση των βάσεων δεδομένων. Πρόκειται για συστήματα όπου τα δεδομένα συλλέγονται, αποθηκεύονται σε ένα εξυπηρετητή κι από ‘κει και πέρα, ένας τουλάχιστον διαχειριστής παρεμβαίνει και εκτελεί τις απαραίτητες ενέργειες. Ωστόσο, η εξέλιξη της τεχνολογίας και η επικράτηση του διαδικτύου, άρχισαν να αποκαλύπτουν κάποιες αδυναμίες. Έχοντας υπόψη ότι οι βάσεις δεδομένων αυτού του χαρακτήρα, στηρίζονται συνήθως σε έναν εξυπηρετητή, στην περίπτωση που αυτός υποστεί κάποια βλάβη και δεν έχει προβλεφθεί κάποιος εφεδρικός που θα τον αντικαταστήσει, όλη η δομή καταρρέει. Επίσης αν κάποιος παρέμβει στα δεδομένα τότε αυτά καταστρέφονται [19].

Για την επίλυση προβλημάτων όπως τα παραπάνω, τα οποία προκύπτουν από συστήματα με πιο συγκεντρωτικό χαρακτήρα, προτείνεται η χρήση της Τεχνολογίας Κατανεμημένου Καθολικού (Distributed Ledger Technology – DLT) που έκανε την εμφάνιση το 2008 όπου ο δημιουργός ή οι δημιουργοί του, με το ψευδώνυμο Satoshi Nakamoto [20], κυκλοφόρησαν την αναφορά ενός ηλεκτρονικού συστήματος νομισμάτων, το blockchain. Την τεχνολογία με την οποία δημιουργήθηκε το πρώτο κρυπτονόμισμα, Bitcoin. Πρόκειται για μια τεχνολογία της οποίας οι δυνατότητες και η προσαρμοστικότητα με τον καιρό κερδίζει όλο και περισσότερο το ενδιαφέρον των επιστημόνων αλλά και της αγοράς. Η τεχνολογία του κατανεμημένου καθολικού και συγκεκριμένα του Blockchain, είναι στην πραγματικότητα ένα συγχρονισμένο μητρώο (ledger) στο όποιο αποθηκεύονται και επαληθεύονται δεδομένα. [21]

Βασικά χαρακτηριστικά που κάνουν το blockchain να ξεχωρίζει είναι, η αποκέντρωση, η αμεταβλητότητα, η διαφάνεια και η ιδιωτικότητα. [22]

Για κάποιους το blockchain θεωρείται μια κατανεμημένη βάση δεδομένων, ωστόσο για άλλους πρόκειται για ένα αποκεντρωμένο δίκτυο που λειτουργεί σαν μητρώο εγγραφών και συναλλαγών που λαμβάνουν χώρα μέσα σ’ αυτό. Ένα ομότιμο δίκτυο (Peer- to- Peer) όπου κάθε χρήστης αποτελεί και έναν κόμβο του. Όσο περισσότεροι κόμβοι υπάρχουν μέσα στο δίκτυο, τόσο καλλιεργείται μεγαλύτερη εμπιστοσύνη και διαφάνεια. Το σύστημα δεν έχει κάποιον κύριο

διαχειριστή. Οι ίδιοι οι χρήστες καλούνται να ελέγξουν και να εγκρίνουν την κάθε ενέργεια. [23] Όλοι οι κόμβοι έχουν τις ίδιες εγγραφές και ενημερώνονται ταυτόχρονα για κάθε αλλαγή μέσα στο μητρώο. Στόχος είναι όλοι να έχουν ακριβώς την ίδια κατάσταση του μητρώου. [24]

Όπως υποδεικνύει η ονομασία του blockchain, πρόκειται για μια αλυσίδα από blocks. Μια διαδοχική αλληλουχία από blocks τα οποία συνδέονται μεταξύ τους με μια κρυπτογραφημένη συμβολοσειρά (hash value), σχηματίζοντας μια αλυσίδα. Το κάθε block περιλαμβάνει, εκτός από τα δεδομένα που ανταλλάσσονται και διακινούνται μέσα στο δίκτυο, μοναδικές πληροφορίες που το διακρίνουν, μια σφραγίδα χρόνου (timestamp) και το hash value. [25] Μια συμβολοσειρά δηλαδή που εκτός ότι ορίζει την θέση του κάθε μπλοκ, έχει ενσωματωμένα κρυπτογραφημένα δεδομένα των συναλλαγών που εμπεριέχονται μέσα στο μπλοκ.[26] Το σημαντικότερο στοιχείο που μπορεί να θεωρηθεί και ο λόγος που ξεχωρίζει σαν τεχνολογία το blockchain, είναι η αμεταβλητότητα του. Κάθε μπλοκ που δημιουργείται, δεν μπορεί να αλλαχθεί, ν' αλλοιωθεί άλλα ούτε και να διαγραφεί. Αυτή η απαραβίαστη ιδιότητα της συγκεκριμένης τεχνολογίας μπορεί να διασφαλίσει την αξιοπιστία και την εγκυρότητα των ενεργειών. [21]

Κατηγορίες Blockchain

Το blockchain με την ιδιαίτερη φύση του, παρατηρήθηκε ότι σαν τεχνολογία είναι αρκετά ευέλικτη κι ενώ ξεκίνησε ως την τεχνολογία που υλοποιήθηκε για την πραγματοποίηση ηλεκτρονικών συναλλαγών, σήμερα αποτελεί βασικό συστατικό σε διάφορους τομείς, όπως η υγεία (healthcare)[27], η εφοδιαστική αλυσίδα (supply chain)[28], η ιατροφαρμακευτική περίθαλψη (medical insurance)[29], το διαδίκτυο των πραγμάτων (IoT)[30] κ.α. Διακρίνεται σε τρείς (3) τύπους. Δημόσιο, ιδιωτικό και η κοινοπραξία των δύο.

Δημόσιο blockchain (public blockchain)

Είναι η πρώτη μορφή του blockchain. Επιτρέπεται η πρόσβαση σε όλους και είναι ένα εντελώς αποκεντρωμένο σύστημα. Όλοι κόμβοι του δικτύου έχουν ίδια δικαιώματα πρόσβασης, επικύρωσης και δημιουργίας block και δεδομένων και μπορούν να συμβάλλουν στον έλεγχο και την συναίνεση μιας συναλλαγής.

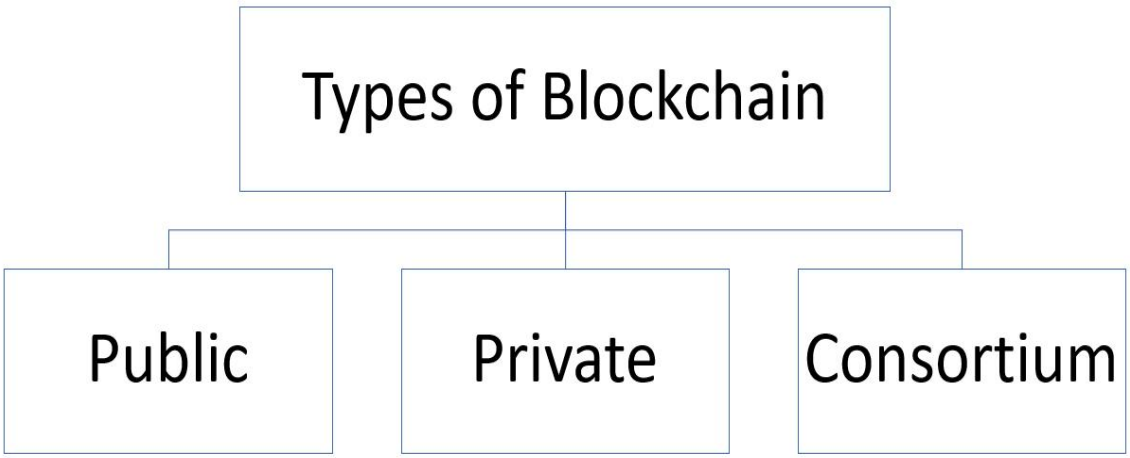
Ιδιωτικό blockchain (private blockchain)

Θα μπορούσε να θεωρηθεί ίσως ο ασφαλέστερος τύπος blockchain. Η ύπαρξη διαχειριστή, δημιουργεί στενά περιθώρια όσο αφορά το ποιος θα έχει πρόσβαση στο blockchain,

καθώς επίσης και το ποιος θα έχει το δικαίωμα ανάγνωσης. Αυτό ακριβώς όμως, είναι και το στοιχείο που συμβάλλει στο δημιουργείται ένα προστατευμένο και ελεγχόμενο περιβάλλον. Η δομή αυτή περιορίζει κατά ένα ποσοστό την αποκέντρωση, που αποτελεί θεμελιώδη χαρακτηριστικό των δημοσίων blockchain, ωστόσο δεν σημαίνει ότι την εξαλείφει.

Κοινοπραξία των δύο (consortium blockchain)

Πρόκειται για μια μορφή που βρίσκει το σημείο περαιτέρω εξέλιξης των blockchain. Πραγματεύεται την ύπαρξη εμπιστοσύνης σε καλύτερα επίπεδα απ' ότι ένα δημόσιο blockchain που είναι ανοικτό σε όλους, προσφέροντας διαχείριση από έναν οργανισμό παρά μιας μεμονωμένης οντότητας χωρίς να διακυβεύεται η εμπιστοσύνη. [31][32]



Εικόνα 2.1 Κατηγορίες Blockchain

Βασικά χαρακτηριστικά του Blockchain

Το blockchain, χωρίς να γίνει εκτενής ανάλυση στην εξέλιξη του μέσα από την χρήση του διάφορους τομείς, διαθέτει κάποια συγκεκριμένα βασικά στοιχεία που συμβάλλουν στην αναγνώριση του ως μια ανατρεπτική και πρωτοπόρα τεχνολογία.

Αποκεντρωμένο (decentralized)

Διακρίνεται περισσότερο σε ένα δημόσιο blockchain, ωστόσο αποτελεί θεμελιώδη στοιχείο της τεχνολογίας γενικότερα. Η αποκεντρωμένη του φύση, προσδίδει στο blockchain την δυνατότητα απαλλαγής από ενδιάμεσους διαχειριστές, μετατρέποντας το σε ένα ασφαλέστερο σύστημα με ίσα δικαιώματα για όλους τους χρήστες.

Διαφάνεια (transparency)

Ακόμα ένα στοιχείο του blockchain που συμβάλει στην αίσθηση της ασφάλειας, είναι η διαφάνεια. Μέσα σε ένα blockchain, όλοι οι χρήστες, κόμβοι του δικτύου, έχουν πρόσβαση στα δεδομένα που υπάρχουν μέσα σε αυτό, καθώς όλοι ενημερώνονται αυτόματα για κάθε ενέργεια που πραγματοποιείται μέσα σ' αυτό.

Αμεταβλητότητα (immutability)

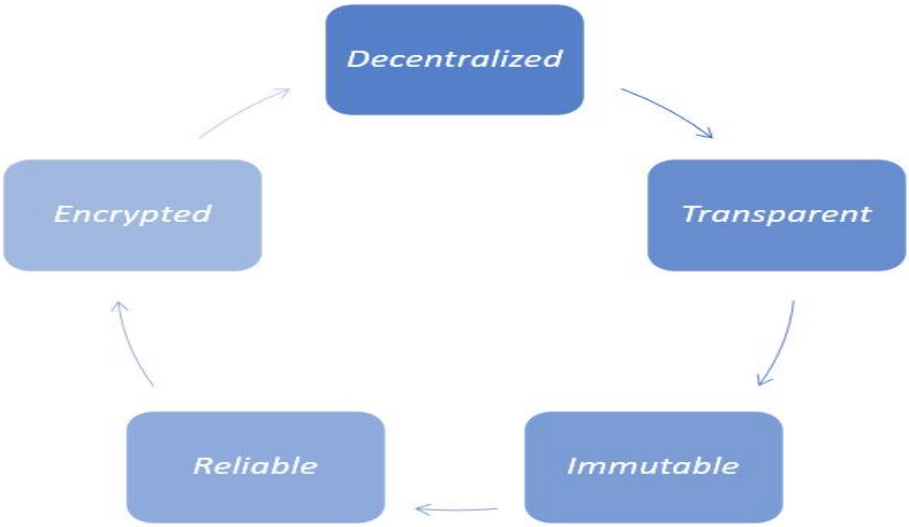
Ίσως το σημαντικότερο χαρακτηριστικό της τεχνολογίας του blockchain. Από την στιγμή που μια συναλλαγή ή όποιας μορφής δεδομένη καταγραφεί στο blockchain, δεν μπορεί να δεχθεί οποιαδήποτε μορφή τροποποίησης, κατά συνέπεια δε μπορεί και να διαγραφεί. [33] [25]

Αξιοπιστία (reliability)

Όλα τα προαναφερθέντα χαρακτηριστικά του blockchain προσδίδουν στην αξιοπιστία της τεχνολογίας καθώς αντίγραφα όλων των εγγραφών, διανέμονται συγχρονισμένα σε όλα τα εμπλεκόμενα μέλη του δικτύου, περιορίζοντας αν όχι εξαλείφοντας κάθε ρίσκο διαφθοράς και αποτυχίας.

Κρυπτογράφηση (encryption)

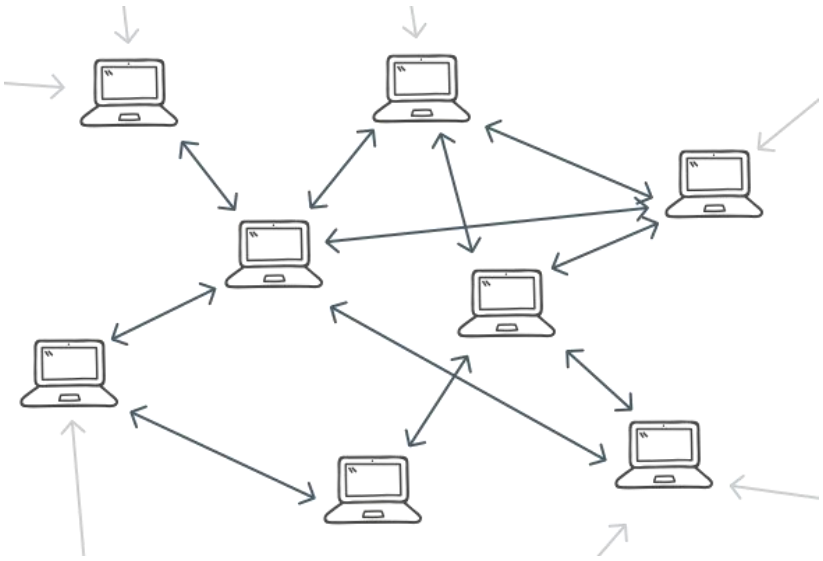
Αναπόσπαστο κομμάτι της τεχνολογίας, καθώς πέραν των συναλλαγών που είναι κρυπτογραφημένες, κάθε block που δημιουργείται και προστίθεται στην αλυσίδα περιλαμβάνει μια κρυπτογραφημένη ταυτότητα που το χαρακτηρίζει, διασφαλίζοντας εκτός από την θέση του, τα περιεχόμενα του. [34]



Εικόνα 2.2 Χαρακτηριστικά του Blockchain

Ομότιμα δίκτυα (Peer-to-peer)

Το πιο διαδεδομένο και γνωστό μοντέλο δικτύου είναι το client- server. Ένας ή περισσότεροι server εξυπηρετούν τους υπόλοιπους clients/ κόμβους του δικτύου, παρέχοντας τις πληροφορίες που ζητούνται, έχοντας απευθείας σύνδεση μαζί τους ή χρησιμοποιώντας άλλους κόμβους μέχρι να φτάσει η πληροφορία στον τελικό χρήστη. Αναφερόμενοι όμως στα ομότιμα δίκτυα, η δομή αλλάζει. Είναι αποκεντρωμένα συστήματα χωρίς κεντρικό διαχειριστή. Όλοι οι χρήστες που απαρτίζουν το δίκτυο, είναι απευθείας συνδεδεμένοι μεταξύ τους. Έχουν ίσα δικαιώματα καθώς και την δυνατότητα επιλογής διαμοιρασμού των πληροφοριών και των υπηρεσιών που έχουν οριστεί να διαθέτουν, επιτρέποντας ή αποκλείοντας την πρόσβαση σε άλλους χρήστες του δικτύου. [35] Η αρχιτεκτονική του δικτύου αλλά και η δυναμική του έχουν σαν αποτέλεσμα μεγαλύτερη ασφάλεια για κάθε χρήστη και κατά συνέπεια για ολόκληρο το δίκτυο. [36]

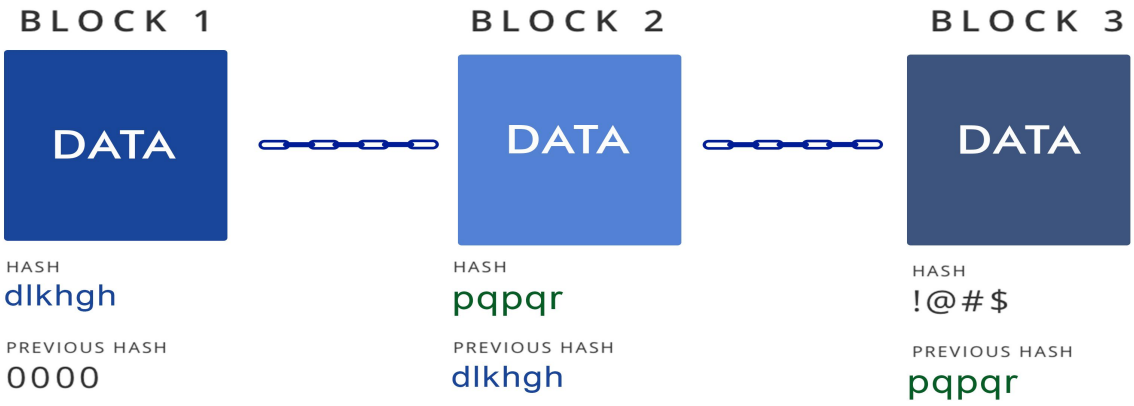


Εικόνα 2.3 Ομότιμο δίκτυο [37]

Block

Το blockchain, είναι αλυσιδωτή δομή αποθήκευσης δεδομένων [38]. Μια χρονολογημένη αλληλουχία συναλλαγών ή δεδομένων συνταγμένα σε μορφή blocks που συνδέονται με μια κρυπτογραφημένη συμβολοσειρά (hash value) που δίνει την αίσθηση ενός είδους "αλυσίδας".

Ο σχηματισμός ενός νέου block, ακολουθεί μετά την επιβεβαίωση από τους χρήστες του δικτύου για την ολοκλήρωση μιας ενέργειας. Αυτόματα λοιπόν το νέο block λαμβάνει χρονοσήμανση και το hash value που θα καθορίσει την θέση του και την μοναδικότητα του. Κάθε block είναι αμετάβλητο και συνδεδεμένο με το προηγούμενο. Η σύνδεση αυτή είναι ουσιαστικά η καταχώρηση του hash value του προηγούμενου block στο νέο (Εικόνα 2.4). Το μέγεθος του block ορίζει και την χωρητικότητα του, συνήθως είναι στα 4 MB.[39]



Εικόνα 2.4 Δομή Blockchain [40]

Κατακερματισμός (Hash)

Η κρυπτογράφηση των πληροφοριών αποτελεί το κυριότερο δομικό στοιχείο της τεχνολογίας του blockchain. Συνήθως περικλείει μια συνάρτηση κατακερματισμού (hash function) και αλγόριθμους ασύμμετρης κρυπτογράφησης. Η ασύμμετρη κρυπτογράφηση χρησιμοποιείται για να αποδειχθεί η κυριότητα του κόμβου μέσω της ψηφιακής υπογραφής, ενώ η hash function για να προστατεύσει την ακεραιότητα των δεδομένων. [40]

Η λειτουργία μιας hash function είναι η μετατροπή μιας μη προσδιορισμένου μεγέθους σειράς δεδομένων ή μεταβλητών, σε ένα προκαθορισμένου μεγέθους συγχωνευμένου αποτελέσματος ή hash value.

Μια *hash function* έχει τα παρακάτω χαρακτηριστικά:

- *Προσδιοριστική:* τα ίδια δεδομένα θα δώσουν το ίδιο αποτέλεσμα όσες φορές κι αν επαναληφθεί η διαδικασία.
- *Αποδοτική:* θα πρέπει να ολοκληρώνει την διαδικασία γρήγορα ώστε να μην επηρεάζει την αποδοτικότητα του συστήματος.
- *Μονόδρομη:* καθιστά ανέφικτο τον προσδιορισμό της εισόδου εκ του αποτελέσματος.
- *Ασυνεχής:* μικρές αλλαγές διαφοροποιούν το αποτέλεσμα.
- *Συγκρουσιακά ανθεκτική:* δεν μπορεί να παραχθεί το ίδιο αποτέλεσμα από διαφορετικές εισόδους. Η κάθε hash value είναι μοναδική. [41]

SHA- 256

Αποτελεί την πιο διαδεδομένη συνάρτηση κρυπτογράφησης κι εκείνη που κυριαρχεί στο blockchain. Η SHA- 256, μέλος της οικογένειας αλγορίθμων κρυπτογράφησης SHA (Secure Hash Algorithm). Αποτελεί μια από ασφαλέστερες και γρήγορες συναρτήσεις κρυπτογράφησης πληρώντας όλες τις προϋποθέσεις που τίθενται ως προς την αποτελεσματικότητα, την αυθεντικότητα και την εγκυρότητα.[42] Λαμβάνει εισόδους οποιουδήποτε μεγέθους και παράγει μοναδικά hash values μεγέθους 256 bits.[43]

Χρονοσήμανση (Timestamp)

Κάθε block κατά την δημιουργία του, λαμβάνει ένα timestamp. Η διαδικασία ξεκινάει από την στιγμή που θα επικυρωθούν από τους χρήστες του δικτύου οι πληροφορίες που θα αποτελέσουν τα περιεχόμενα του νέου block.[44] Το timestamp ουσιαστικά υποδεικνύει τον χρόνο που έγινε το block δεκτό από το δίκτυο. Στην πραγματικότητα όμως, πρόκειται για την απόδοση του μέσου όρου του χρόνου που έκαναν οι χρήστες του δικτύου να επιβεβαιώσουν την πληροφορία. Ωστόσο γνωρίζοντας ότι η διαδικασία απόδοσης ενός timestamp σε ένα block δεν μπορεί να είναι ακριβής, έχει δοθεί το περιθώριο ενός με δύο ωρών για να θεωρείται αξιόπιστη. Συγκεκριμένα αυτό συμβαίνει διότι, παρά το γεγονός ότι όλοι οι κόμβοι είναι χρονικά συγχρονισμένοι, υπάρχει διαφορά ανάμεσα στην τοπική ώρα του χρήστη και την παγκόσμια ώρα UTC. Για να προκύψει το timestamp, σε πρώτο επίπεδο θα συνδυαστούν οι ώρες επιβεβαίωσης των χρηστών και στην συνέχεια ο τοπικός χρήστης υπολογίζει την διαφορά ανάμεσα σε τοπική και παγκόσμια ώρα. Αυτό ουσιαστικά κάνει το κάθε timestamp μοναδικό και αδύνατον να επαναληφθεί συμβάλλοντας έτσι στο να αποφευχθεί ο κίνδυνος επαναλαμβανόμενης καταχώρησης.[43] [45]

Συναλλαγή (Transaction)

Γνωρίζοντας ότι πλέον η τεχνολογία του blockchain δεν χρησιμοποιείται μόνο για συστήματα κρυπτονομισμάτων αλλά έχει ξεκινήσει να εφαρμόζεται σε διάφορούς τομείς, ο όρος συναλλαγή αποκτά μια πιο γενικευμένη έννοια. Η επιβεβαιωμένη, από τους χρήστες του δικτύου, καταχώρηση δεδομένων που οδηγεί στην δημιουργίας ενός νέου block, θεωρείται συναλλαγή. Στην ευρύτερη αυτή σημασιολογία του όρου, συμβάλλει και η δημιουργία νέων τύπων blockchain καθώς και η προσαρμοστικότητα της τεχνολογίας έχει διαφοροποιήσει την σημερινή της μορφή από την αρχική.

Ωστόσο θα ήταν πολύ βοηθητικό να αναφέρουμε την βασική δομή του blockchain και την πορεία μια συναλλαγής:

1. Αποστέλλεται η συναλλαγή ανάμεσα σε όλους του κόμβους ενός ομότιμου δικτύου.
2. Οι χρήστες/ κόμβοι του δικτύου επιβεβαιώνουν την συναλλαγή και την κατάσταση του χρήστη μέσα από συγκεκριμένους αλγόριθμους.
3. Με την ολοκλήρωση της επιβεβαίωσης δημιουργείται το νέο block δεδομένων στο blockchain το οποίο είναι μόνιμο και αμετάβλητο.
4. Η συναλλαγή ολοκληρώθηκε [31]

Κεφάλαιο 3- Θεμελίωση Επικοινωνίας με το CNIL PIA

Παρουσίαση Προβλήματος

Ο μηχανισμός της ΕΑΠΔ, εκτός από την δυνατότητα να προφυλάξει τα προσωπικά δεδομένα των Ευρωπαίων πολιτών, διασφαλίζει τους οργανισμούς που σχεδιάζουν και κατασκευάζουν τις υπηρεσίες και τα προϊόντα που επεξεργάζονται τα δεδομένα προσωπικού χαρακτήρα. Ωστόσο, ενώ αρκετοί οργανισμοί ανά τον κόσμο, ανέπτυξαν πρότυπα και εφαρμογές που βοηθούν τους ενδιαφερόμενους στην υλοποίηση μιας ΕΑΠΔ, δεν φαίνεται να υπάρχει μέριμνα για το σημαντικότερο ίσως στοιχείο, την διασφάλιση μιας ΕΑΠΔ μετά την ολοκλήρωση της. Το πρόβλημα λοιπόν που πραγματεύεται αυτή η εργασία έχει να κάνει με το

εγχείρημα και την ανεύρεση του τρόπου εκείνου που θα μπορέσει να εξασφαλιστεί η αυθεντικότητα μιας ΕΑΠΔ.

Για την υλοποίηση της εργασίας επιλέχθηκε η εφαρμογή CNIL PIA, η οποία αποτελεί το λογισμικό που προσφέρει την δυνατότητα επεξεργασίας και προσαρμογής σύμφωνα με τα κριτήρια που έχουμε θέσει. Ενώ για την επίτευξη του στόχου, που αφορά στην διασφάλιση της μη επεξεργασίας, οποιαδήποτε μορφής μιας ΕΑΠΔ, επιλέχθηκε η τεχνολογία blockchain. Η τεχνολογία που φαίνεται να έχει κερδίσει το ενδιαφέρον της αγοράς με την ευελιξία της αλλά κυρίως με το σημαντικότερο χαρακτηριστικό της, την αμεταβλητότητα των δεδομένων που καταχωρούνται σε αυτό.

Εφαρμογή CNIL PIA

Ο CNIL παρέχει την εφαρμογή σε δύο μορφές, Portable και Web. Σ' αυτό το σημείο θα πρέπει να αναφερθεί ότι η Εκτίμηση Αντικτύπου για τα Προσωπικά Δεδομένα (ΕΑΠΔ) στην εφαρμογή αναφέρεται ως PIA (Privacy Impact Assessment).

Portable Έκδοση

Η portable έκδοση μπορεί να εγκατασταθεί σε προσωπικό υπολογιστή. Είναι συμβατή με λειτουργικά συστήματα: Windows, Linux και Mac OS, χωρίς να αναφέρονται ιδιαίτερες απαιτήσεις υλικού (hardware). Το περιβάλλον της εφαρμογής είναι απλό και φιλικό προς τον χρήστη, οργανωμένο σε ενότητες και πεδία. Στην διάθεση του χρήστη, διατίθεται ένα ολιγόλεπτο βίντεο καθοδήγησης και γνωριμίας με το περιβάλλον αλλά και αναλυτικά εγχειρίδια χρήσης με παραδείγματα, τα οποία μπορούν να κατατοπίσουν σε ικανοποιητικό βαθμό ακόμα και έναν αρχάριο. Εκτός αυτών, μέσα στην εφαρμογή, σε κάθε πεδίο, με την ενεργοποίηση του, εμφανίζεται μια αυτόματα προσαρμοζόμενη βοηθητική στήλη επεξηγώντας το περιεχόμενο του. Όταν ολοκληρωθεί η διαδικασία, η εφαρμογή CNIL PIA, προσφέρει την εξαγωγή της εκάστοτε PIA σε τρεις τύπους αρχείων:

- JSON, παρέχοντας την περίπτωση περαιτέρω επεξεργασίας σε μεταγενέστερη περίοδο,
- PDF, όπου αποτυπώνεται η τελική της μορφή, και
- DOCX, σε περίπτωση πιστοποίησης ή αυθεντικοποίησης της με τρόπους και μεθόδους που επιλέγει ο χρήστη.

Web Έκδοση

Στην web έκδοση, το λογισμικό αναφέρεται περισσότερο σε επιχειρήσεις και οργανισμούς που έχουν συγκεκριμένες απαιτήσεις και δυνατότητες. Απαιτείται εξυπηρετητής (server), ενώ διαθέτει frontend και backend λειτουργία. Πρόκειται για μια εφαρμογή ανοικτού κώδικα. Το λειτουργικό σύστημα που προτείνεται είναι το Ubuntu, ενώ οι απαιτήσεις που θα πρέπει να πληροί το υπολογιστικό σύστημα, είναι: επεξεργαστή(CPU): i5, Μνήμη (RAM): 4GBκαι διαθέσιμη χωρητικότητα τοπικού δίσκου: 20GB.

Ο CNIL, έχει δημιουργήσει ένα δημόσιο λογαριασμό στο GitHub, όπου και προσφέρει όλα όσα είναι απαραίτητα για την έναρξη χρήσης του λογισμικού. Εγχειρίδια πληροφοριών εγκατάστασης, πληροφορίες υποστήριξης, ενημέρωση σχετικά με την δομή, αρχεία και υπερσυνδέσμους, τα οποία καθοδηγούν τον χρήστη βήμα προς βήμα για την χρήση του λογισμικού, τόσο για το frontend όσο και για το backend. Όλα παρέχονται στην ιστοσελίδα του οργανισμού. [46]

REST (Representational State Transfer- Μεταφορά Αντιπροσωπευτικής Κατάστασης)

Το REST είναι μια αρχιτεκτονική που εμπεριέχει ένα σύνολο κανόνων με στόχο την βελτίωση του διαδικτύου, επιδιώκοντας καλύτερη απόδοση, επεκτασιμότητα, απλότητα, δυνατότητα τροποποίησης, ορατότητα, φορητότητα και αξιοπιστία. Υιοθετείται κατά κόρον για την δημιουργία web APIs. Οι εφαρμογές που ακολουθούν στους κανόνες αυτούς ονομάζονται RESTful APIs. Το HTTP (Hyper Text Transfer Protocol- Πρωτόκολλο Μεταφοράς Υπερκειμένου), είναι το πρωτόκολλο που έχει επικρατήσει και σ’ αυτή την περίπτωση, να χρησιμοποιείται, όπου ένας client, μέσα από τις αντίστοιχες μεθόδους, μπορεί να δημιουργήσει, να ανακτήσει, να ανανεώσει και να διαγράψει δεδομένα, χρησιμοποιώντας XML ή JSON για την μεταφορά τους.

Θεμελιώδης Αρχές του REST

Εξυπηρετητής - Πελάτης (Client - Server): στο μοντέλο client- server, υπάρχει σαφής διαχωρισμός ρόλων και έτσι ο client δεν εμπλέκεται σε θέματα αποθήκευσης όπως και ο server δεν σχετίζεται με ευθύνες της interfaces (interface) client. Κατά αυτόν τον τρόπο βελτιώνεται η

φορητότητα (portability) του user interface σε περισσότερες πλατφόρμες καθώς και η επεκτασιμότητα (scalability), μέσα από την απλοποίηση των στοιχείων του server.

Statelessness: τα stateless πρωτόκολλα δίνουν την δυνατότητα στον server να λαμβάνει μηνύματα που δεν εμπεριέχουν όλη την πληροφορία από την έναρξη της επικοινωνίας με τον client. Ωστόσο, μπορεί να αντιλαμβάνεται το μήνυμα ανεξάρτητα από τα προηγούμενα. Η ενέργεια αυτή συμβάλλει στην μείωση της συμφόρησης του server, βελτιώνοντας την απόδοση του.

Cacheability: οι servers θα πρέπει να ξεκαθαρίζουν άμεσα αν διαθέτουν κρυφή μνήμη (cache) ή όχι, στοχεύοντας στην πρόληψη και αποτροπή των clients από το να παρέχουν, σε απαντήσεις επόμενων αιτημάτων, παλιότερα ή λανθασμένα δεδομένα. Χρησιμοποιώντας σωστά την ύπαρξη κρυφής μνήμης, μπορούν να μειωθούν οι συνδιαλλαγές μεταξύ client- server βελτιώνοντας σημαντικά την επεκτασιμότητα και την απόδοση.

Layered system: η ύπαρξη ή μη απευθείας σύνδεσης του client με τον τελικό server, δεν επηρεάζει την μεταξύ τους επικοινωνία. Η ύπαρξη, ενδιάμεσων servers μπορεί να συμβάλλει στην επεκτασιμότητα, επιτυγχάνοντας ομαλοποίηση του φόρτου είτε παρέχοντας διαμοιραζόμενη κρυφή μνήμη, είτε όχι. Επιπλέον, ενσωματώνεται ένα πρόσθετο επίπεδο ασφάλειας πάνω από τις υπηρεσίες ιστού. Πλεονέκτημα επίσης αποτελεί ότι οι ενδιάμεσοι servers μπορούν να ζητήσουν από πολλαπλάσιους servers να δημιουργήσουν απαντήσεις σε έναν client.

Code on demand: πρόκειται για προαιρετικό περιορισμό που ο server μπορεί να παρατείνει ή να προσαρμόσει την λειτουργικότητα του client μεταφέροντας εκτελέσιμο κώδικα.

Uniform interface: ο πιο καθοριστικός περιορισμός του REST. Απλοποιεί και διαχωρίζει όλα τα τμήματα της αρχιτεκτονικής, επιδιώκοντας ανεξαρτησία. Διαθέτοντας:

- Αναγνώριση των διαδικτυακών πόρων μέσα στα αιτήματα.
- Δυνατότητα επεξεργασίας των διαδικτυακών πόρων μέσα από την αναπαράσταση του.
- Αιτήματα που περιέχουν αρκετή πληροφορία ώστε να μπορούν να περιγράψουν τα ίδια πως θα πρέπει να προωθηθούν.
- Hypermedia as the engine of application state (HATEOAS)[47]

API (Application Programming Interface- Διεπαφή Προγραμματισμού Εφαρμογών)

Πρόκειται για δομικά στοιχεία λογισμικού που συμβάλλουν στην ανάπτυξη επικοινωνίας ανάμεσα σε δύο διαδικτυακές εφαρμογές. Ένα σύνολο πρωτοκόλλων και εργαλείων τα οποία παρέχουν την δυνατότητα κατασκευής νέων εφαρμογών με την χρήση ήδη υπαρκτών εφαρμογών, χωρίς να χρειάζεται η γνώση των τεχνικών λεπτομερειών του κάθε συστήματος.

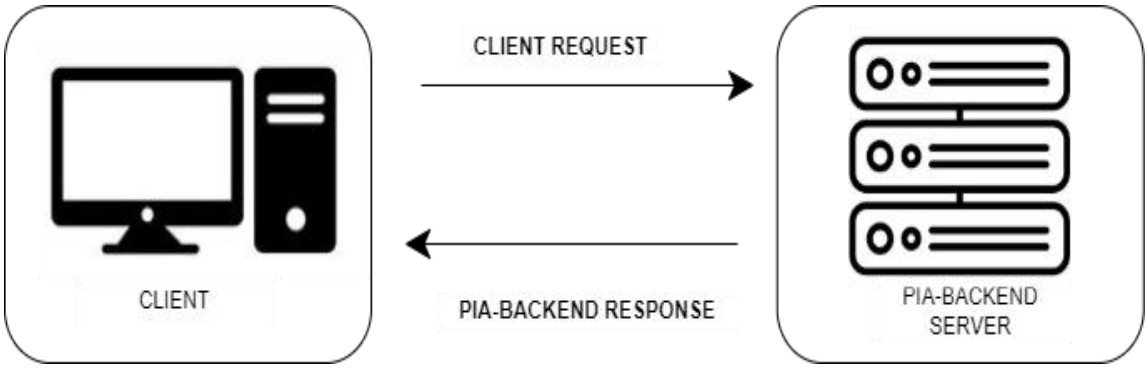
Χαρακτηριστικά:

- Πληρούν τα πρότυπα (κυρίως HTTP και REST), τα οποία είναι φιλικά προς τους χρήστες.
- Προτιμώνται ως υπηρεσίες παρά ως κωδικός προγράμματος.
- Η χρήση τους απευθύνεται σε συγκεκριμένο κοινό, συνήθως εκείνους που ασχολούνται με την ανάπτυξη λογισμικού.
- Δεδομένης της τυποποιημένης ταυτότητας τους, προσφέρουν ισχυρότερη πειθαρχία στην ασφάλεια και τη διαχείριση.
- Υπάρχει κλιμάκωση των API για να μπορούν να εξυπηρετούνται περισσότεροι χρήστες.
- Διαθέτουν δικό τους κύκλο ανάπτυξης λογισμικού σχεδιασμού, δοκιμών, κατασκευής, διαχείρισης και διαχείρισης εκδόσεων.[48]

Μέθοδοι HTTP

Το HTTP (Hyper Text Transfer Protocol- Πρωτόκολλο Μεταφοράς Υπερκειμένου) είναι το πιο γνωστό πρωτόκολλο επικοινωνίας στον ιστό. Είναι υπεύθυνο για την μεταφορά δεδομένων ανάμεσα σε έναν client (πελάτη) και έναν server (διακομιστή) μέσα από συγκεκριμένες μεθόδους. Οι βασικές κατηγορίες κι αυτές που θα χρησιμοποιηθούν σ' αυτή την εργασία είναι:

- GET: χρησιμοποιείται για την ανάκληση πόρων.
- POST: δημιουργεί έναν πόρο.
- PATCH: ανανεώνει το περιεχόμενο ενός πόρου που υπάρχει ήδη.
- DELETE: διαγράφει έναν υπάρχον πόρο.[49]



Εικόνα 3.1 Μοντέλο client- server

Κάθε φορά που εκτελείται μια μέθοδος από τον client, ουσιαστικά αποστέλλεται ένα αίτημα (request), στον server. Η επιτυχημένη ή μη ολοκλήρωση του αιτήματος υποδηλώνεται όταν ο server επιστρέψει σαν απάντηση (response) έναν συγκεκριμένο κωδικό κατάστασης (status code). Βασικοί κωδικοί απαντήσεων: [50]

Κωδικός	**Περιγραφή**
200 (OK)	Απάντηση επιτυχούς αιτήματος
201 (Created)	Επιτυχής δημιουργία πόρου
204 (No content)	Επιτυχή αίτηση χωρίς σώμα απάντησης
400 (Bad request)	Λανθασμένη διατύπωση αιτήματος
404 (Not found)	Δεν βρέθηκε ο πόρος
500 (Internal server error)	Απροσδόκητη αποτυχία

Πίνακας 3.1 Κωδικοί απαντήσεων

Swagger

Βασικό εργαλείο για την περιγραφή ενός RESTful API. Πρόκειται για ένα αναλυτικά συνταγμένο κείμενο που επεξηγεί κάθε πεδίο της εκάστοτε εφαρμογής. Μέσα από το swagger ο χρήστης μπορεί να καθοδηγηθεί για το πως θα χρησιμοποιήσει την εφαρμογή. Για παράδειγμα,

μέσα από το swagger ο χρήστης λαμβάνει την πληροφορία για την ορθή σύνταξη μια HTTP μεθόδου. Η δομή του φαίνεται στην Εικόνα 2.2. [51]

```yaml
swagger: "2.0"
info:
  version: 1.0.0
  title: "PIA back"
  description: An API to manage PIA (Privacy Impact Assesment)
  termsOfService: http://swagger.io/terms/
  contact:
    name: Atnos API Team
    email: contact@atnos.com
    url: https://www.atnos.com
  license:
    name: MIT
    url: http://github.com/gruntjs/grunt/blob/master/LICENSE-MIT
host: pia-back.atnosapp.com
basePath: /
schemes:
  - https
consumes:
  - multipart/form-data
produces:
  - application/json
paths:
  /pias:
    get:
      description: Returns all PIAs.
      operationId: listPias
      parameters:
        - name: export
          in: query
```

Εικόνα 3.2 PIA Swagger

JSON (JavaScript Object Notation)

Είναι ένα format αρχείου για ανταλλαγή δεδομένων. Εύκολο για τους ανθρώπους να το διαβάσουν και για τις μηχανές να το αναλύσουν και να το παράγουν. Η δομή του διαμορφώνεται σε ζεύγη, όπου μια άνω κάτω τελεία ξεχωρίζει το αριστερό μέρος, το κλειδί από το δεξί μέρος, της τιμής. Για παράδειγμα:

“όνομα” : ” Μαρία”

Όσα ζεύγη απαρτίζουν το JSON, μεταξύ τους διαχωρίζοντα με κόμμα (“,”). Οι τύποι δεδομένων που μπορούν να αποτελέσουν τιμές ενός ονόματος είναι: συμβολοσειρά (string), αριθμός (number), δυαδική τιμή (Boolean),κενό (Null), αντικείμενο (Object) και Πίνακας (Array).

Εικόνα 3.3 PIA JSON

Ανάπτυξη Λογισμικού

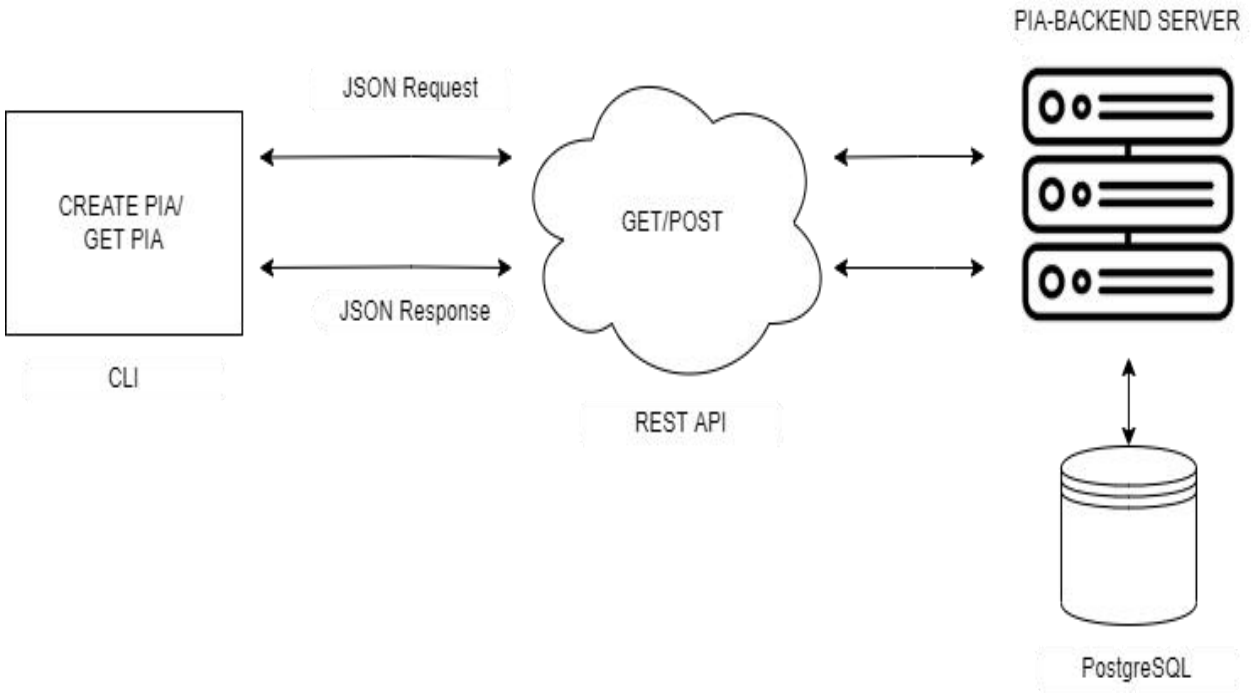
Επικοινωνία με την CNIL PIA

Ακολουθώντας όσα ορίζονται από το REST API μέσα από python scripts, διαμορφωμένα σύμφωνα με την δομή των HTTP μεθόδων και το swagger της εφαρμογής CNIL PIA, πραγματώνεται η επικοινωνία ανάμεσα στον client και τον PIA backend server. Η επικοινωνία θα ολοκληρωθεί έπειτα από την επιβεβαιωμένη επιτυχία όλων των μεθόδων για κάθε τομέα της εφαρμογής (Πίνακας 3.2.). Για να γίνει σαφή η δομή που θα ακολουθηθεί θα πρέπει να σημειωθεί, πως αρκετοί από τους τομείς της CNIL PIA, εξαρτώνται από κάποιους άλλους. Για παράδειγμα, ο τομέας comments με τον τομέα PIA. Ουσιαστικά αυτό που συμβαίνει είναι ότι αν πρώτα δεν δημιουργηθεί μια PIA, δεν μπορεί να δημιουργηθεί κι ένα comment, δεδομένου ότι τα comments δημιουργούνται για μια PIA. Με την ίδια λογική λειτουργούν και οι HTTP μέθοδοι. Για να μπορέσει να διαγραφεί μια PIA, θα πρέπει πρώτα να έχει δημιουργηθεί. Οπότε η μέθοδος POST θα πρέπει να προπορεύεται της μεθόδου DELETE.

Πεδίο	Λειτουργία
PIA	Δημιουργία, ανάκληση, ανανέωση και διαγραφή μιας PIA
Example	Ανάκληση αποθηκευμένου παραδείγματος PIA (αν είναι διαθέσιμο)
Import	Εισαγωγή μιας PIA
Duplicate	Δημιουργία διπλότυπου μιας υπάρχουσας PIA
Answer	Δημιουργία, ανάκληση, ανανέωση και διαγραφή απαντήσεων σε συγκεκριμένη PIA
Evaluations	Δημιουργία, ανάκληση, ανανέωση και διαγραφή αξιολόγησης σε συγκεκριμένη PIA
Measures	Δημιουργία, ανάκληση, ανανέωση και διαγραφή μέτρων σε συγκεκριμένη PIA
Attachments	Δημιουργία, ανάκληση, ανανέωση και διαγραφή επισυναπτόμενων αρχείων σχετιζόμενα με τα μέτρα σε συγκεκριμένη PIA
Structures	Δημιουργία, ανάκληση, ανανέωση και διαγραφή δομής για συγκεκριμένη PIA

Πίνακας 3.2 Τομείς CNIL PIA

Ξεκινώντας, ο client στέλνει μήνυμα (request) στον PIA Backend server, επιδιώκοντας να λάβει πίσω σαν απάντηση (response) το αντίστοιχο status code που θα επιβεβαιώσει ότι το αίτημα ολοκληρώθηκε επιτυχώς.



Εικόνα 3.4 Επικοινωνία με CNIL PIA

Δημιουργία μιας PIA

POST: η σύνταξη του request απαιτεί το URL που αντιστοιχεί στην βάση δεδομένων της εφαρμογής όπου και θα δημιουργηθεί η PIA και στο JSON που περιέχει τα περιεχόμενα των παραμέτρων όπως τις έχουμε ορίσει. Με την αποστολή του request, θα πρέπει να ληφθεί το status code από τον PIA backend server που θα επιβεβαιώνει την επιτυχή δημιουργία της PIA. Η επιβεβαίωση αυτή είναι και ο πρώτος έλεγχος επικοινωνίας.

```
r = requests.post(url, json=pia)
```

Η δημιουργία μιας PIA συνεπάγεται και την απόδοση ενός αναγνωριστικού κωδικού (id). Ο κωδικός αυτός διακρίνει την κάθε PIA και αποτελεί καθοριστικό παράγοντα για μετέπειτα την επεξεργασία της.

Ανάκληση μιας PIA

GET: Η εφαρμογή δίνει την δυνατότητα δύο ειδών ανάκλησης. Πρώτον, να ανακαλέσει ο χρήστης το σύνολο των PIA. Σε αυτή την περίπτωση η σύνταξη ενός ξεχωριστού GET request δεν κρίνεται αναγκαίο και χρήσιμο να εκτελείται ανεξάρτητα καθώς θα αποτελέσει βασικό στοιχείο για την υλοποίηση των υπολοίπων μεθόδων. Δεύτερον, να ανακαλέσει ο χρήστης μια

συγκεκριμένη PIA. Γνωρίζοντας τον αναγνωριστικό κωδικό (id) της, συντάσσεται το GET request με το συγκεκριμένο id και το προκαθορισμένο URL. Όπως και στην μέθοδο POST, αιτείται η επιστροφή του αντίστοιχου κωδικού απάντησης που θα καθορίσει αν το αίτημα ολοκληρώθηκε επιτυχώς ή όχι.

```
r = requests.get(url + "/" + str(pia["id"]))
```

Ανανέωση μιας PIA

PATCH: η μέθοδος αυτή συντάσσεται όπως η μέθοδος POST με την μόνη διαφορά ότι ουσιαστικά αντικαθιστά το JSON μιας συγκεκριμένης PIA. Όπως και στις προηγούμενες περιπτώσεις με την αποστολή του request αναμένεται η λήψη του αντίστοιχου κωδικού απάντησης ώστε να επιβεβαιωθεί ότι ολοκληρώθηκε επιτυχώς η ανανέωση.

```
r = requests.patch(url + "/" + str(pia["id"]), json=payload)
```

Διαγραφή μιας PIA

DELETE: η σύνταξη της μεθόδου DELETE έχει την ίδια δομή με την μέθοδο GET και περιλαμβάνει το id και το URL της PIA. Ο κωδικός απάντησης θα μας επιβεβαιώσει την διαγραφή της συγκεκριμένης PIA.

```
r = requests.delete(url + "/" + str(pia["id"]))
```

Με βάση τις παραπάνω HTTP μεθόδους και τον Πίνακα 3.2 θεμελιώνεται και ολοκληρώνεται η επικοινωνία με την CNIL PIA. Κατά αυτόν τον τρόπο πλέον είναι διαθέσιμες όλες οι δυνατότητες που προσφέρει η εφαρμογή.

CNIL PIA και Blockchain

Γνωρίζοντας την δυναμικότητα του blockchain σαν τεχνολογία, σε συνδυασμό με την ευελιξία και την προσαρμοστικότητα του, επιλέξαμε να το χρησιμοποιήσουμε σαν μέσω αποθήκευσης και προστασίας των PIA που θα συντάσσονται μέσω του της εφαρμογής CNIL PIA. Έτσι λοιπόν στην περίπτωση μας, όταν συμβαίνει, ο όρος συναλλαγή θα αναφέρεται στην

δημιουργία μιας νέας PIA. Πριν ξεκινήσει η ανάλυση της δομής του προσαρμοσμένου στις ανάγκες της συγκεκριμένης εργασίας blockchain, θα πρέπει να αναφέρουμε ότι πρόκειται για ένα ιδιωτικό μητρώο στο οποίο ο διαχειριστής είναι ο μόνος που μπορεί να δημιουργεί την πληροφορία, στην συγκεκριμένη περίπτωση την PIA η οποία στην συνέχεια θα οδηγήσει την δημιουργία ενός νέου block.

Για την ανάπτυξη ενός blockchain, η διαδικασία ξεκινάει με την δημιουργία μιας class που θα περιλαμβάνει τα δομικά στοιχεία, βάση των οποίων θα συσταθούν στην συνέχεια οι μέθοδοι που θα εκτελούν διαδοχικά τις ενέργειες εκείνες που θα κατασκευάζουν κάθε φορά το νέο block το οποίο θα προστίθεται στην αλυσίδα, blockchain.

Όπως έχουμε περιγράψει παραπάνω, το πρώτο πράγμα που πρέπει να οριστεί είναι η δημιουργία ενός block, ενώ ακολουθεί η επεξεργασία των συναλλαγών που θα καταχωρηθούν σε αυτό και τέλος η προσθήκη μιας hash value. Οπότε με βάση αυτή την δομή, δημιουργείται σε πρώτο επίπεδο μια class με τις παρακάτω μεταβλητές:

- **chain**: για αρχή θα είναι μια άδεια λίστα, που στην συνέχεια θα προστίθενται τα blocks. Ουσιαστικά θα είναι το blockchain.
- **pendingTransactions:** εκεί θα στέλνονται οι συναλλαγές (στην περίπτωση μας η PIA) μέχρι την στιγμή που θα καταχωρηθούν σε ένα block.
- **newBlock:** η μέθοδος που θα διαμορφωθεί παρακάτω στον κώδικα και θα είναι ουσιαστικά εκείνη που θα προσθέτει το νέο block στην αλυσίδα, blockchain.

Κατόπιν, εφόσον κατασκευάσαμε μια άδεια αλυσίδα, ακολουθεί η δημιουργία της συνάρτησης που θα δημιουργεί το νέο block με συγκεκριμένα περιεχόμενα τύπου JSON και θα προσθέτει στην αλυσίδα το νέο block. Το JSON περιέχει τα παρακάτω:

- **index:** ώστε να υπάρχει μια αύξουσα αναφορά που θα προσδιορίζει πόσα blocks υπάρχουν στο blockchain. Σε κάθε νέο block θα προστίθεται συν ένα.
- **timestamp:** η χρονοσήμανση που θα δηλώνει την ώρα δημιουργίας του block.
- **transactions:** οι συναλλαγές που υπάρχουν στην λίστα αναμονής και μπορούν να καταχωρηθούν στο block.
- **previous hash:** μια εκδοχή από το hash value του τελευταίου block.

Ακολουθεί η συνάρτηση που προσθέτει κάθε νέα συναλλαγή στην λίστα αναμονής μέχρι να προχωρήσει στο επόμενο βήμα που είναι η καταχώρηση σε νέο block που θα μεγαλώσει την αλυσίδα κατά ένα. Στην συνάρτηση αυτή, προσδιορίζεται:

- **ο αποστολέας,**
- **παραλήπτης και**

- **το αντικείμενο** (στην περίπτωση μας η PIA)

Τέλος, ο βασικός κορμός του κώδικα ανάπτυξης ενός blockchain, ολοκληρώνεται με συνάρτηση που θα κρυπτογραφεί το περιεχόμενο του νέου block. Εδώ θα χρησιμοποιηθεί η συνάρτηση SHA -256. [52]

Διεπαφή Γραμμής Εντολών (Command Line Interface- CLI)

Αφορά μια από τις επιλογές όπου ένας χρήστης μπορεί να αλληλοεπιδράσει με μια συσκευή ή ένα λογισμικό. Πρόκειται για μια δομή εντολών με συγκεκριμένο συντακτικό, οι οποίες διατυπώνονται και εκτελούνται μέσω ενός παράθυρου τερματικού. Η λειτουργία του CLI είναι σχετικά απλή, καθώς με την σύνταξη της κατάλληλης εντολής, κατά την εκτέλεση της ο χρήστης λαμβάνει πίσω μια απάντηση από το σύστημα.

Βασικά χαρακτηριστικά ενός CLI :

- Είναι ακριβής
- Καλύτερος έλεγχος
- Δεν απαιτεί ιδιαίτερους πόρους
- Έχει μεγάλη απόδοση
- Επεκτασιμότητα
- Απαιτεί εμπειρία για την χρήση του
- Η μορφή του είναι λιτή[53]

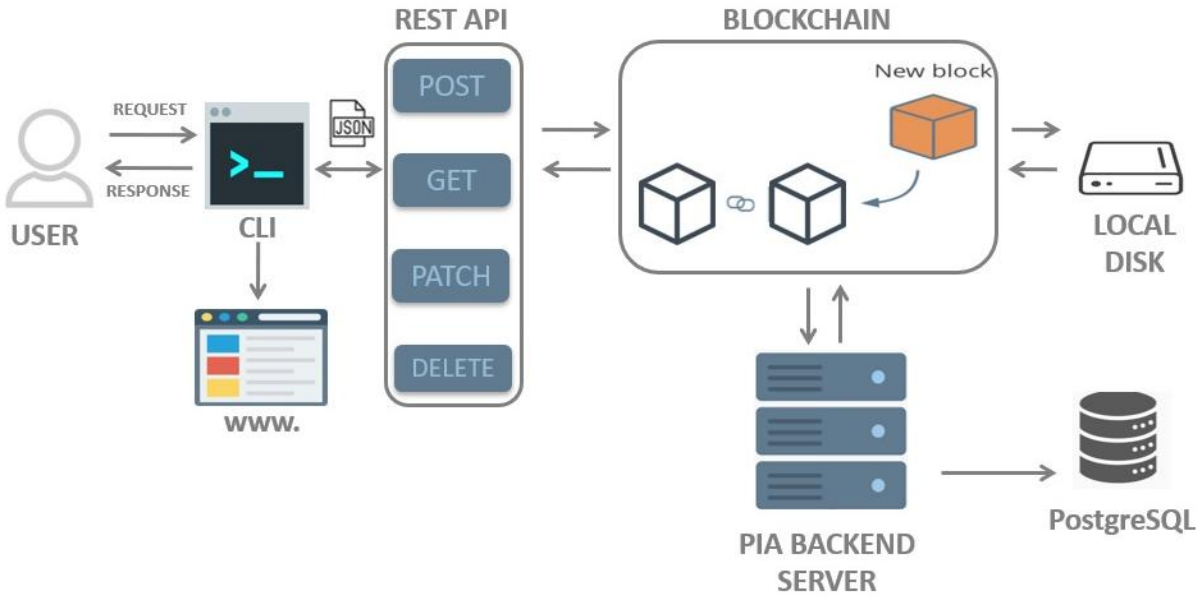
Ανάπτυξη CLI

Ο σχεδιασμός και η ανάπτυξη ενός CLI, απαιτεί συνδυαστικά με την γνώση του υπεύθυνου ανάπτυξης λογισμικού, την βέλτιστη αξιοποίηση των δεδομένων που παρέχονται από την εκάστοτε εφαρμογή. Κατά κύριο λόγο το CLI, είναι το προγραμματιστικό εργαλείο εκείνο που πραγματοποιεί την καταχώρηση και την παρουσίαση πληροφοριών μέσα από εντολές διατυπωμένες σύμφωνα με ένα συγκεκριμένο συντακτικό. Οι εντολές αυτές βασίζονται πάνω σε κριτήρια που ονομάζονται arguments (args). Έναν συγκεκριμένο τύπο δεδομένων μέσω των οποίων παρουσιάζονται όσα θέλουμε να καταχωρήσουμε στο σύστημα.

Στο CLI που δημιουργήθηκε για την συγκεκριμένη εφαρμογή, οι παράμετροι που περιλαμβάνονται σε κάθε πεδίο της CNIL PIA, αποτελούν τα arguments που θα τεθούν για την δόμηση κάθε ενέργειας. Στην δημιουργία μιας PIA για παράδειγμα, όσες παράμετροι περιέχονται στο JSON (Εικόνα 2.3), είναι τα εν δυνάμει arguments. Για την βέλτιστη αξιοποίηση των δυνατοτήτων της εφαρμογής το swagger αποτελεί το εργαλείο εύκολης αναζήτησης και προσδιορισμού κάθε λεπτομέρειας αναφορικά με τις παραμέτρους και τα πεδία της εφαρμογής.

Η υλοποίηση ενός CLI μπορεί να εξελιχθεί σε ένα πολυεπίπεδο σύστημα. Στην παρούσα εργασία, αυτό θα γίνει ιδιαίτερα αισθητό, καθώς η εφαρμογή που επιδιώκουμε να χρησιμοποιήσουμε, αποτελείται από αρκετά πεδία. Αποσκοπώντας στην αποτελεσματικότητα, επιλέξαμε να γίνει ομαδοποίηση των επιλογών ανά πεδίο, δεδομένου ότι έχουν κοινά χαρακτηριστικά. Για παράδειγμα η ομαδοποίηση της δημιουργίας, ανάκληση, ανανέωση και διαγραφή μιας PIA. Στόχος είναι η κατηγοριοποίηση, κατά ένα τρόπο, των πεδίων ώστε να μπορέσουν να αποτυπωθούν με μεγαλύτερη ακρίβεια τα κριτήρια, καλύπτοντας όλες τις δυνατότητες που παρέχονται. Στην συνέχεια συντάσσονται, ανά πεδίο, οι παράμετροι σε arguments.

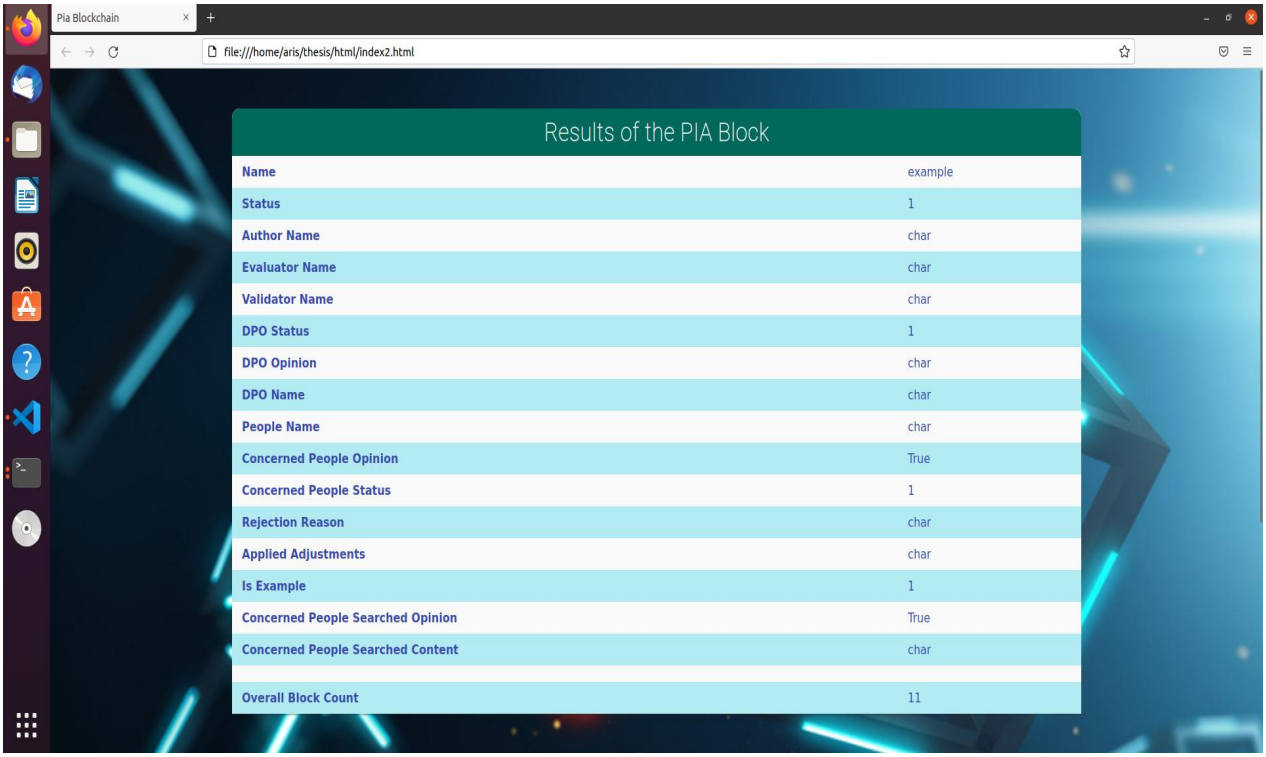
Κατόπιν, συντάσσεται ένα ενδιάμεσο script στο οποίο περιέχονται αναλυτικά διατυπωμένες όλες οι ενέργειες που έχει την δυνατότητα να εκτελέσει ο χρήστης. Έχοντας αυτό σαν βάση, με την συμβολή ενός βοηθητικού πίνακα γίνεται η άντληση των ενεργειών οι οποίες στην συνέχεια, ανάλογα με το είδος της ενέργειας, πραγματοποιεί τις σχετικές αλλαγές στο blockchain.



Εικόνα 3.5 Εξέλιξη CNIL PIA με την χρήση της τεχνολογίας Blockchain

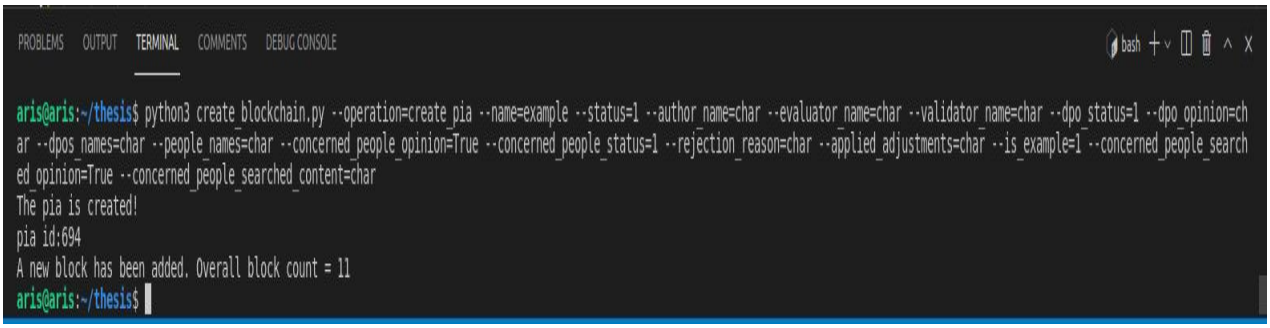
Παρουσίαση

Όπως αναφέραμε, η χρήση ενός CLI σημαίνει και επιλογή απεικόνισης και διαχείρισης των εντελών μέσω ενός παράθυρου τερματικού. Ένα μη γραφικό περιβάλλον. Στην συγκεκριμένη εργασία, κατά την υλοποίηση του CLI, επιλέχθηκε να δημιουργηθεί ένα html αρχείο (Εικόνα 4.1.) το οποίο ανανεώνεται αυτόματα κάθε φορά που θα δημιουργείται μια νέα PIA, απεικονίζοντας τα αποτελέσματα σε ένα πιο φιλικό περιβάλλον.



Εικόνα 3.6 Παρουσίαση Δημιουργίας μιας PIA

Ωστόσο, δεδομένου ότι αναφερόμαστε σε μια εφαρμογή που κύριο στόχο της έχει την αποθήκευση του αποτελέσματος της PIA σε ένα block, ενός ιδιωτικού blockchain, θεωρήθηκε φρόνιμο να δημιουργηθεί στον κώδικα ένας μετρητής που αυτόματα καταμετρά και ενημερώνει τον χρήστη τον συνολικό αριθμό των blocks που έχουν δημιουργηθεί μέχρι εκείνη την στιγμή. Στην Εικόνα 3.7, αποτυπώνεται η εντολή δημιουργίας μιας PIA μέσω του CLI, με τις προαπαιτούμενες παραμέτρους όπως φαίνονται στο JSON της CNIL PIA (Εικόνα 3.3), καθώς και την απάντηση ότι η PIA δημιουργήθηκε, το pia_id που την συνοδεύει αλλά και το συνολικό αριθμό blocks που απαρτίζουν το blockchain. Ο συνολικός αριθμός εμφανίζεται επίσης και στην Εικόνα 3,6.



Εικόνα 3.7 Εκτέλεση ενός αιτήματος δημιουργίας PIA

Κεφάλαιο 4- Συμπεράσματα

Στην παρούσα εργασία, κύριο μέλημα μας ήταν η δημιουργία ενός συστήματος που θα εξασφάλιζε την αυθεντικότητα μιας PIA που δημιουργείται με την εφαρμογή του CNIL. Για να επιτευχθεί αυτός ο στόχος χρησιμοποιήθηκε η τεχνολογία κατανεμημένου καθολικού (DLT) και συγκεκριμένα η τεχνολογία blockchain. Δημιουργήθηκε ένα ιδιωτικό blockchain, στο οποίο έχει προγραμματιστεί να αποθηκεύεται αυτόματα σε ένα νέο block, κάθε νέα PIA. Με αυτό τον τρόπο λύνουμε το πρόβλημα της διαβλητότητας. Μελλοντικά, θα μπορούσαμε να ενσωματώσουμε την παραπάνω πρόταση, προσαρμόζοντας κατάλληλα την εφαρμογή CNIL PIA.

Βιβλιογραφία

[1] Ο. Ε. ΣΕΒ, "Ο Νέος Γενικός Κανονισμός Για Την Προστασία Δεδομένων (Gdpr)," 2018.

[2] S. Wachter, "The GDPR and the internet of things: A three-step transparency model," *Law, Innov. Technol.*, vol. 10, no. 2, pp. 266–294, 2018, doi: 10.1080/17579961.2018.1527479.

[3] Κ. Σιασιάκος, Σ. Αναστασίου, and Κ. Τούντας, "Εκτίμηση Των Επιπτώσεων Σχετικά Με Την Προστασία Δεδομένων Σε Έργα Ηλεκτρονικής Διακυβέρνησης," *Εκπαίδευση, Δια Βίου Μάθηση, Έρευνα Και Τεχνολογική Ανάπτυξη, Καινοτομία Και Οικονομία*, vol. 1, p. 542, 2017, doi: 10.12681/elrie.817.

[4] "Privazyplan." https://www.privacy-regulation.eu/el/4.htm (accessed Jan. 20, 2022).

[5] U. Borchers, "European Commission," 2014. http://ec.europa.eu/education/policy/higher-education/bologna-process_bg (accessed Jan. 20, 2022).

[6] D. Kloza *et al.*, "Data protection impact assessment in the European Union: developing a template for a report from the assessment process," no. 1, pp. 1–52, 2020, doi: 10.31228/osf.io/7qrfp.

[7] I. UK, "What is a DPIA?" https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/data-protection-impact-assessments-dpias/what-is-a-dpia/ (accessed Jan. 22, 2022).

[8] Ε. Επιτροπή, "Πότε πρέπει να γίνεται εκτίμηση αντίκτυπου σχετικά με την προστασία δεδομένων (ΕΑΠΔ);" https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/when-data-protection-impact-assessment-dpia-required_el (accessed May 29, 2022).

[9] ICO-InformationCommissioner's Office, "How do we do a DPIA?" https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/data-protection-impact-assessments-dpias/how-do-we-do-a-dpia/ (accessed May 29, 2022).

[10] Γ. Ε. Δ. Π. Χαρακτήρα, "Υπεύθυνος Προστασίας Δεδομένων (DPO)." https://www.dataprotection.gov.cy/dataprotection/dataprotection.nsf/page2b_gr/page2b_gr?opendocument (accessed May 29, 2022).

[11] K. Demetzou, "Data Protection Impact Assessment: A tool for accountability and the unclarified concept of 'high risk' in the General Data Protection Regulation," *Comput. Law Secur. Rev.*, vol. 35, no. 6, pp. 1–14, 2019, doi: 10.1016/j.clsr.2019.105342.

[12] I. Consulting, “Recital 75Risks to the Rights and Freedoms of Natural Persons*.” https://gdpr-info.eu/recitals/no-75/.

[13] I. C. O. (ICO), “Conducting privacy impact assessments code of practice,” *Ico.Org.Uk*, pp. 1–55, 2014, [Online]. Available: https://ico.org.uk/media/for-organisations/documents/1595/pia-code-of-practice.pdf%0Ahttps://ico.org.uk/for_organisations/guidance_index/~/media/documents/library/Data_Protection/Practical_application/pia-code-of-practice-final-draft.pdf.

[14] O. of the P. C. (OPC) N. Zealand, “Privacy impact assessment toolkit,” 2019. https://www.privacy.org.nz/publications/guidance-resources/privacy-impact-assessment/ (accessed Apr. 14, 2021).

[15] O. of the A. I. Commissioner, “Guide to undertaking privacy impact assessments,” 2014. https://www.oaic.gov.au/privacy/guidance-and-advice/guide-to-undertaking-privacy-impact-assessments/ (accessed Apr. 14, 2021).

[16] F. C. N. de l’Informatique et des Liberte, “Privacy Impact Assessment (PIA),” 2018. https://www.cnil.fr/en/PIA-privacy-impact-assessment-en (accessed Apr. 15, 2021).

[17] G. of Canada, “Directive on Privacy Impact Assessment,” 2020. https://www.tbs-sct.gc.ca/pol/doc-eng.aspx?id=18308 (accessed Apr. 15, 2021).

[18] International Organization for Standardization, “ISO/IEC 29134 information technology – security techniques – privacy impact assessment – guidelines,” 2017. https://www.iso.org/obp/ui/#iso:std:iso-iec:29134:ed-1:v1:en (accessed Apr. 14, 2021).

[19] N. Singh and M. Vardhan, “Distributed Ledger Technology based Property Transaction System with Support for IoT Devices,” *Int. J. Cloud Appl. Comput.*, vol. 9, no. 2, pp. 60–78, 2019, doi: 10.4018/ijcac.2019040104.

[20] Wikipedia, “Satoshi Nakamoto,” 2022. https://en.wikipedia.org/wiki/Satoshi_Nakamoto (accessed Mar. 10, 2022).

[21] M. Finck, *Blockchains and the General Data Protection Regulation*, no. July. 2018.

[22] M. Bower, C. Howe, N. McCredie, A. Robinson, and D. Grover, “Augmented Reality in education - cases, places and potentials,” *EMI. Educ. Media Int.*, vol. 51, no. 1, pp. 1–15, 2014, doi: 10.1080/09523987.2014.889400.

[23] Wikipedia, “Bitcoin,” 2022. https://en.wikipedia.org/wiki/Bitcoin (accessed Mar. 05, 2022).

[24] S. Seebacher and R. Schüritz, “Blockchain technology as an enabler of service systems: A structured literature review,” *Lect. Notes Bus. Inf. Process.*, vol. 279, pp. 12–23, 2017, doi: 10.1007/978-3-319-56925-3_2.

[25] M. R. Biktimirov, A. V. Domashev, P. A. Cherkashin, and A. Y. Shcherbakov, “Blockchain Technology: Universal Structure and Requirements,” *Autom. Doc. Math. Linguist.*, vol. 51, no. 6, pp. 235–238, 2017, doi: 10.3103/s0005105517060036.

[26] Y. Ren, Y. Leng, F. Zhu, J. Wang, and H. J. Kim, “Data storage mechanism based on blockchain with privacy protection in wireless body area network,” *Sensors (Switzerland)*, vol. 19, no. 10, pp. 1–16, 2019, doi: 10.3390/s19102395.

[27] A. Shahnaz, U. Qamar, and A. Khalid, “Using Blockchain for Electronic Health Records,” *IEEE Access*, vol. 7, pp. 147782–147795, 2019, doi: 10.1109/ACCESS.2019.2946373.

[28] M. Alazab, S. Alhyari, A. Awajan, and A. B. Abdallah, “Blockchain technology in supply chain management: an empirical study of the factors affecting user adoption/acceptance,” *Cluster Comput.*, vol. 4, no. 2019, 2020, doi: 10.1007/s10586-020-03200-4.

[29] L. Zhou, L. Wang, and Y. Sun, “MIStore: a Blockchain-Based Medical Insurance Storage System,” *J. Med. Syst.*, vol. 42, no. 8, 2018, doi: 10.1007/s10916-018-0996-4.

[30] L. Tseng, L. Wong, S. Otoum, M. Aloqaily, and J. Ben Othman, “Blockchain for Managing Heterogeneous Internet of Things: A Perspective Architecture,” *IEEE Netw.*, vol. 34, no. 1, pp. 16–23, 2020, doi: 10.1109/MNET.001.1900103.

[31] M. Niranjanamurthy, B. N. Nithya, and S. Jagannatha, “Analysis of Blockchain technology: pros, cons and SWOT,” *Cluster Comput.*, vol. 22, no. 2, pp. 14743–14757, 2019, doi: 10.1007/s10586-018-2387-5.

[32] F. & L. LLP, “Types of Blockchain: Public, Private, or Something in Between,” 2022. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between (accessed Mar. 24, 2022).

[33] A. R. Golosova, Julija, “the advantages and disadvantages of the Inclusion.....,” *2018 IEEE 6th Work. Adv. Information, Electron. Electr. Eng.*, p. 50, 2018.

[34] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, “Everything You Wanted to Know about the Blockchain: Its Promise, Components, Processes, and Problems,” *IEEE Consum. Electron. Mag.*, vol. 7, no. 4, pp. 6–14, 2018, doi: 10.1109/MCE.2018.2816299.

[35] B. Yang and H. Garcia-Molina, “Yang, B., & Garcia-Molina, H. (2002). Improving

Search in Peer-to-Peer Systems. Proceedings of the 22nd International Conference on Distributed Computing Systems ({ICDCS}).," *Yang, B., Garcia-Molina, H. (2002). Improv. Search Peer-to-Peer Syst. Proc. 22nd Int. Conf. Distrib. Comput. Syst. ({ICDCS})*., p. 1, 2002.

[36] N. J. Alpern and R. J. Shimonski, "Network Fundamentals," *Elev. Hour Network+*, pp. 1–18, 2010, doi: 10.1016/b978-1-59749-428-1.00003-5.

[37] P. Think, "[BlockChain] go realize BlockChain 7: Network," 2022. https://programmer.ink/think/blockchain-go-realize-blockchain-7-network.html (accessed Mar. 15, 2022).

[38] H. Wang and Y. Song, "Secure Cloud-Based EHR System Using Attribute-Based Cryptosystem and Blockchain," *J. Med. Syst.*, vol. 42, no. 8, 2018, doi: 10.1007/s10916-018-0994-6.

[39] C. Staff, "How a Block in the Bitcoin Blockchain Works," 2022. https://www.gemini.com/cryptopedia/what-is-block-in-blockchain-bitcoin-block-size (accessed Mar. 23, 2022).

[40] J. Fu, S. Qiao, Y. Huang, X. Si, B. Li, and C. Yuan, "A Study on the Optimization of Blockchain Hashing Algorithm Based on PRCA," *Secur. Commun. Networks*, vol. 2020, 2020, doi: 10.1155/2020/8876317.

[41] A. Rosic, "What Is Hashing? [Step-by-Step Guide-Under Hood Of Blockchain]," 2020. https://blockgeeks.com/guides/what-is-hashing/ (accessed Mar. 27, 2022).

[42] R. Kashyap, K. Arora, A. Azam, and M. Sharma, "Immutable and Privacy Protected E-Certificate Repository on Blockchain," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 3, pp. 1631–1635, 2020, doi: 10.35940/ijeat.c5256.029320.

[43] G. Kalyani and S. Chaudhari, *Survey on 6LoWPAN Security Protocols in IoT Communication*, vol. 601. 2020.

[44] D. O. Jaquet-Chiffelle, E. Casey, and J. Bourquenoud, "Tamperproof timestamped provenance ledger using blockchain technology," *Forensic Sci. Int. Digit. Investig.*, vol. 33, p. 300977, 2020, doi: 10.1016/j.fsidi.2020.300977.

[45] B. Academy, "What is Timestamp?" https://academy.bit2me.com/en/blockchain-timestamp/ (accessed Mar. 26, 2022).

[46] C. N. de l'Informatique et des Libertés, "The open source PIA software helps to carry out

data protection impact assessment.” https://www.cnil.fr/en/open-source-pia-software-helps-carry-out-data-protection-impact-assesment (accessed Mar. 22, 2022).

[47] Wikipedia, “Representational state transfer (REST),” 2022. https://en.wikipedia.org/wiki/Representational_state_transfer (accessed Feb. 15, 2022).

[48] W. Hoogenraad, “Διεπαφή Προγραμματισμού Εφαρμογών,” 2018. https://el.itpedia.nl/2018/11/02/wat-zijn-apis-application-programming-interface (accessed Feb. 03, 2022).

[49] Wikipedia, “Πρωτόκολλο Μεταφοράς Υπερκειμένου (HTTP),” 2021. https://el.wikipedia.org/wiki/Πρωτόκολλο_Μεταφοράς_Υπερκειμένου (accessed Feb. 05, 2022).

[50] Bunny.net, “Typical HTTP Response Codes.” https://support.bunny.net/hc/en-us/articles/360024887131-Typical-HTTP-Response-Codes?/?pk_campaign=DynamicAD&pk_source=DynamicAD&pk_medium=DynamicAD&pk_keyword=&device=c&gclid=CjwKCAiAyPyQBhB6EiwAFUuakk2Mn_WVUxhfRohVnSMaojQ1dBLiIJLXMO4kl5dvR6m-mr9jK1w76Bo (accessed Feb. 25, 2022).

[51] Wikipedia, “Swagger (Software),” 2022. https://en.wikipedia.org/wiki/Swagger_(software)] (accessed Feb. 10, 2022).

[52] JavaTpoint, “Building a Blockchain using Python.” https://www.javatpoint.com/building-a-blockchain-using-python (accessed Mar. 15, 2022).

[53] Gadget-info, “Διαφορά μεταξύ CLI και GUI,” 2019. https://el.gadget-info.com/difference-between-cli (accessed Mar. 15, 2022).

Appendixes

Appendix A - Επικοινωνία με CNIL PIA Backend

PIA

```python
import requests


def create_pia(name):

    url = "http://127.0.0.1:3000/pias"
    pia = {"name": name}

    r = requests.post(url, json=pia)
    return r.status_code


def test_create_pia():
    assert create_pia("prime") == 201
    assert create_pia("prime1") == 201
    assert create_pia("prime2") == 201
```

PIA ID

```python
import requests, json


def get_pia():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]))
        assert r.status_code == 200
    return r.status_code


def test_get_pia():
    assert get_pia() == 200

```

```python
import requests, json


def patch_pias():

    url = "http://127.0.0.1:3000/pias"
    payload = {"name": "oliver"}
    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.patch(url + "/" + str(pia["id"]), json=payload)
        assert r.status_code == 200
    return r.status_code


def test_patch_pias():
    assert patch_pias() == 200

```

```python
import requests, json


def delete_pias():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        r = requests.delete(url + "/" + str(pia["id"]))
        assert r.status_code == 204
    return r.status_code


def test_delete_pias():
    delete_pias()

```

PIA EXAMPLE

```python
import requests


def get_example():
    url = "http://127.0.0.1:3000/pias/example"

    r = requests.get(url)
    print(r.text)
    assert r.status_code == 200
    return r.status_code


def test_get_example():
    assert get_example() == 200

```

PIA DUPLICATE

```python
import requests, json


def create_duplicate():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.post(url + "/" + str(pia["id"]) + "/duplicate")
        assert r.status_code == 200
    return r.status_code


def test_duplicate():
    assert create_duplicate() == 200

```

PIA IMPORT

```python
import requests


def import_pia():

    url = "http://127.0.0.1:3000/pias/import"
    files = {
        "data": ("pia.json", open("pia.json", "rb")),
    }

    r = requests.post(url, files=files)
    print(r.text)
    return r.status_code


def test_import_pia():
    assert import_pia() == 200

```

PIA ANSWERS

```python
import requests, json


def get_answers():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]) + "/answers")
        assert r.status_code == 200
    return r.status_code


def test_get_answers():
    assert get_answers() == 200

```

```python
import requests, json


def post_answers():

    url = "http://127.0.0.1:3000/pias"
    answers = {"pia_id": 1, "reference_to": "string"}
    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        answers["pia_id"] = int(pia["id"])
        r = requests.post(url + "/" + str(pia["id"]) + "/answers", json=answers)
        assert r.status_code == 201
    return r.status_code


def test_post_answers():
    assert post_answers() == 201

```

```python
import requests, json


def get_answers_id():
    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/answers")
        answers = json.loads(res.text)
        response_1 = 0
        for answer in answers:
            response = requests.get(
                url + "/" + str(pia["id"]) + "/answers" + "/" + str(answer["id"])
            )
            assert response.status_code == 200
            response_1 = response.status_code
        return response_1


def test_get_answers_id():
    assert get_answers_id() == 200

```

```python
import requests, json


def patch_answers_id():
    url = "http://127.0.0.1:3000/pias"
    payload = {"pia_id": 1, "reference_to": "any"}

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        payload["pia_id"] = int(pia["id"])
        res = requests.get(url + "/" + str(pia["id"]) + "/answers")
        answers = json.loads(res.text)
        resp_1 = 0
        for answer in answers:
            resp = requests.patch(
                url + "/" + str(pia["id"]) + "/answers" + "/" + str(answer["id"]),
                json=payload,
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_patch_answers_id():
    assert patch_answers_id() == 200

```

```python
import requests, json


def delete_answers_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/answers")
        answers = json.loads(res.text)

        for answer in answers:
            resp = requests.delete(
                url + "/" + str(pia["id"]) + "/answers" + "/" + str(answer["id"])
            )
            assert resp.status_code == 204
            return resp.status_code


def test_delete_answers_id():
    assert delete_answers_id() == 204

```

PIA COMMENTS

```python
import requests, json


def get_comments():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]) + "/comments")
        assert r.status_code == 200
    return r.status_code


def test_get_comments():
    assert get_comments() == 200

```

```python
import requests, json


def post_comments():

    url = "http://127.0.0.1:3000/pias"
    comments = {
        "pia_id": 0,
        "description": "string",
        "reference_to": "string",
        "for_measure": True,
    }

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.post(url + "/" + str(pia["id"]) + "/comments", json=comments)
        assert r.status_code == 201
    return r.status_code


def test_post_comments():
    assert post_comments() == 201

```

```python
import requests, json


def get_comments_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/comments")
        comments = json.loads(res.text)
        resp_1 = 0
        for comment in comments:
            resp = requests.get(
                url + "/" + str(pia["id"]) + "/comments" + "/" + str(comment["id"])
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_get_comments_id():
    assert get_comments_id() == 200

```

```python
import requests, json


def patch_comments_id():
    url = "http://127.0.0.1:3000/pias"
    payload = {
        "pia_id": 0,
        "description": "string",
        "reference_to": "string",
        "for_measure": True,
    }

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        payload["pia_id"] = int(pia["id"])
        res = requests.get(url + "/" + str(pia["id"]) + "/comments")
        comments = json.loads(res.text)
        resp_1 = 0
        for comment in comments:
            resp = requests.patch(
                url + "/" + str(pia["id"]) + "/comments" + "/" + str(comment["id"]),
                json=payload,
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_patch_commnets_id():
    assert patch_comments_id() == 200

```

```python
import requests, json


def delete_comments_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/comments")
        comments = json.loads(res.text)
        resp_1 = 0
        for comment in comments:
            resp = requests.delete(
                url + "/" + str(pia["id"]) + "/comments" + "/" + str(comment["id"])
            )
            assert resp.status_code == 204
            resp_1 = resp.status_code
        return resp_1


def test_delete_comments_id():
    assert delete_comments_id() == 204

```

PIA EVALUATIONS

```python
import requests, json


def get_evaluations():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]) + "/evaluations")
        assert r.status_code == 200
    return r.status_code


def test_get_evaluations():
    assert get_evaluations() == 200

```

```python
import requests, json


def post_evaluations():

    url = "http://127.0.0.1:3000/pias"
    payload = {"name": "prime", "reference_to": "string"}

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.post(url + "/" + str(pia["id"]) + "/evaluations", json=payload)
        assert r.status_code == 201
    return r.status_code


def test_post_evaluations():
    assert post_evaluations() == 201

```

```python
import requests, json


def get_evaluations_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/evaluations")
        evaluations = json.loads(res.text)
        resp_1 = 0
        for evaluation in evaluations:
            resp = requests.get(
                url
                + "/"
                + str(pia["id"])
                + "/evaluations"
                + "/"
                + str(evaluation["id"])
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_get_evaluations_id():
    assert get_evaluations_id() == 200

```

```python
import requests, json


def patch_evaluations_id():
    url = "http://127.0.0.1:3000/pias"
    payload = {"name": "prime", "reference_to": "string"}

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/evaluations")
        evaluations = json.loads(res.text)
        resp_1 = 0
        for evaluation in evaluations:
            resp = requests.patch(
                url
                + "/"
                + str(pia["id"])
                + "/evaluations"
                + "/"
                + str(evaluation["id"]),
                json=payload,
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_patch_evaluations_id():
    assert patch_evaluations_id() == 200

```

```python
import requests, json


def delete_evaluations_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/evaluations")
        evaluations = json.loads(res.text)
        resp_1 = 0
        for evaluation in evaluations:
            resp = requests.delete(
                url
                + "/"
                + str(pia["id"])
                + "/evaluations"
                + "/"
                + str(evaluation["id"])
            )
            assert resp.status_code == 204
            resp_1 = resp.status_code
        return resp_1


def test_delete_evaluations_id():
    assert delete_evaluations_id() == 204

```

PIA MEASURES

```python
import requests, json


def get_measures():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]) + "/measures")
        assert r.status_code == 200
    return r.status_code


def test_get_measures():
    assert get_measures() == 200

```

```python
import requests, json


def post_measures():

    url = "http://127.0.0.1:3000/pias"
    payload = {"name": "proto"}

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.post(url + "/" + str(pia["id"]) + "/measures", json=payload)
        assert r.status_code == 201
    return r.status_code


def test_post_measures():
    assert post_measures() == 201

```

```python
import requests, json


def get_measures_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/measures")
        measures = json.loads(res.text)
        response_1 = 0
        for measure in measures:
            response = requests.get(
                url + "/" + str(pia["id"]) + "/measures" + "/" + str(measure["id"])
            )
            assert response.status_code == 200
            response_1 = response.status_code
        return response_1


def test_get_measures_id():
    assert get_measures_id() == 200

```

```python
import requests, json

def patch_measures_id():
    url = "http://127.0.0.1:3000/pias"
    payload = {"name": "proto"}

    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/measures")
        measures = json.loads(res.text)
        resp_1 = 0
        for measure in measures:
            resp = requests.patch(url + "/" + str(pia["id"]) + "/measures" + "/" + str(measure["id"]), json = payload)
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1

def test_patch_measures_id():
    assert patch_measures_id() == 200
```

```python
import requests, json


def delete_measures_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/measures")
        measures = json.loads(res.text)
        resp_1 = 0
        for measure in measures:
            resp = requests.delete(
                url + "/" + str(pia["id"]) + "/measures" + "/" + str(measure["id"])
            )
            assert resp.status_code == 204
            resp_1 = resp.status_code
        return resp_1


def test_delete_measures_id():
    assert delete_measures_id() == 204

```

PIA ATTACHMENTS

```python
import requests, json


def get_attachments():

    url = "http://127.0.0.1:3000/pias"

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        r = requests.get(url + "/" + str(pia["id"]) + "/attachments")
        assert r.status_code == 200
    return r.status_code


def test_get_attachmnets():
    assert get_attachments() == 200

```

```python
import requests, json


def post_attachments():

    url = "http://127.0.0.1:3000/pias"
    payload = {
        "attachment": {
            "pia_id": 0,
            "description": "a",
            "reference_to": "a",
            "for_measure": True,
            "attached_file": {
                "pia_id": 0,
                "file": "g",
                "name": "test.txt",
                "mime_type": "text/plain",
            },
        }
    }

    r = requests.get(url)
    pias = json.loads(r.text)
    for pia in pias:
        payload["attachment"]["pia_id"] = int(pia["id"])
        payload["attachment"]["attached_file"]["pia_id"] = int(pia["id"])
        r = requests.post(url + "/" + str(pia["id"]) + "/attachments", json=payload)
        print(r.status_code == 200)
        # assert r.status_code == 200
    return r.status_code


def test_post_attachments():
    assert post_attachments() == 200

```

```python
import requests, json


def get_attachments_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/attachments")
        attachments = json.loads(res.text)
        resp_1 = 0
        for attachment in attachments:
            resp = requests.get(
                url
                + "/"
                + str(pia["id"])
                + "/attachments"
                + "/"
                + str(attachment["id"])
            )
            assert resp.status_code == 200
            resp_1 = resp.status_code
        return resp_1


def test_get_attachments_id():
    assert get_attachments_id() == 200

```

```python
import requests, json


def delete_attachments_id():

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)
    pias = json.loads(r.text)

    for pia in pias:
        res = requests.get(url + "/" + str(pia["id"]) + "/attachments")
        attachments = json.loads(res.text)
        resp_1 = 0
        for attachment in attachments:
            resp = requests.delete(
                url
                + "/"
                + str(pia["id"])
                + "/attachments"
                + "/"
                + str(attachment["id"])
            )
            assert resp.status_code == 204
            resp_1 = resp.status_code
        return resp_1


def test_delete_attachments_id():
    assert delete_attachments_id() == 204

```

PIA STUCTURES

```python
import requests


def get_structures():

    url = "http://127.0.0.1:3000/structures"

    r = requests.get(url)
    assert r.status_code == 200
    return r.status_code


def test_get_structures():
    assert get_structures() == 200

```

```python
import requests


def post_structures(name):

    url = "http://127.0.0.1:3000/structures"
    payload = {"structure": {"name": name, "sector_name": "m", "data": "11111111"}}
    r = requests.post(url, json=payload)
    return r.status_code


def test_post_structures():
    assert post_structures("nouvelle") == 201
    assert post_structures("nouvelle1") == 201
    assert post_structures("nouvelle2") == 201
```

```python
import requests, json


def get_structures_id():

    url = "http://127.0.0.1:3000/structures"

    r = requests.get(url)
    structures = json.loads(r.text)
    for structure in structures:
        r = requests.get(url + "/" + str(structure["id"]))
        assert r.status_code == 200
    return r.status_code


def test_get_structures_id():
    assert get_structures_id() == 200
```

```python
import requests, json


def patch_structures_id():

    url = "http://127.0.0.1:3000/structures"
    body = {"name": "struct"}

    r = requests.get(url)
    structures = json.loads(r.text)
    for structure in structures:
        r = requests.get(url + "/" + str(structure["id"]), json=body)
        assert r.status_code == 200
    return r.status_code


def test_patch_structures_id():
    assert patch_structures_id() == 200

```

```python
import requests, json


def delete_structures_id():

    url = "http://127.0.0.1:3000/structures"

    r = requests.get(url)
    structures = json.loads(r.text)
    for structure in structures:
        r = requests.delete(url + "/" + str(structure["id"]))
        assert r.status_code == 204
    return r.status_code


def test_delete_structures_id():
    assert delete_structures_id() == 204

```

Appendix B - Blockchain

```python
import hashlib
import json
from time import time

class Blockchain(object):

    def __init__(self):
        self.chain = []
        self.pending_transactions = []
        self.count = 0
        self.new_block(previous_hash="Dpia_test")


    def new_block(self, previous_hash=None):
        block = {
            "index": len(self.chain) + 1,
            "timestamp": time(),
            "transactions": self.pending_transactions,
            "transaction_id": self.count,
            "previous_hash": previous_hash or self.hash(self.chain[-1]),
        }
        self.count += 1
        self.pending_transactions = []
        self.chain.append(block)


        return block

    def persist(self, path):
        with open(path, "w") as f:
            json.dump(self.chain, f)

    def unpersist(self, path):
        with open(path, "r") as f:
            blocks = json.load(f)
            for block in blocks:
                self.chain.append(block)
```

```
39
40
41      @property
42      def last_block(self):
43
44          return self.chain[-1]
45
46
47
48      def new_transaction(self, sender, recipient, pia):
49          transaction = {"sender": sender, "recipient": recipient, "pia": pia}
50          self.pending_transactions.append(transaction)
51          return self.last_block["index"] + 1
52
53
54
55      def hash(self, block):
56          string_object = json.dumps(block, sort_keys=True)
57          block_string = string_object.encode()
58
59          raw_hash = hashlib.sha256(block_string)
60          hex_hash = raw_hash.hexdigest()
61
62          return hex_hash
63
```

Appendix C - CLI

Ομαδοποίηση ανά τομέα και ανάλυση επιλογών με χρήση βοηθητικού πίνακα

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--pia_id",
        type=int,
        default=0,
        help="Which is the pia_id?",
    )
    parser.add_argument(
        "--name",
        type=str,
        default=" ",
        help="What's the name of the object the pia refers to?",
    )
    parser.add_argument(
        "--status", type=int, default=0, help="What's the status of the pia?"
    )
    parser.add_argument(
        "--author_name",
        type=str,
        default=" ",
        help="What's the name of the pia author?",
    )
    parser.add_argument(
        "--evaluator_name",
        type=str,
        default=" ",
        help="What's the name of the pia evaluator?",
    )
```

```python
    parser.add_argument(
        "--validator_name",
        type=str,
        default=" ",
        help="What's the name of the pia validator?",
    )
    parser.add_argument(
        "--dpo_status", type=int, default=0, help="What's the dpo's status?"
    )
    parser.add_argument(
        "--dpo_opinion", type=str, default=" ", help="What's the dpo's option?"
    )
    parser.add_argument(
        "--dpos_names", type=str, default=" ", help="What's the dpo's name?"
    )
    parser.add_argument(
        "--people_names", type=str, default=" ", help="What's the people name?"
    )
    parser.add_argument(
        "--concerned_people_opinion",
        type=str,
        default=" ",
        help="What's concerned people opinion?",
    )
    parser.add_argument(
        "--concerned_people_status",
        type=int,
        default=0,
        help="What's concerned people status?",
    )
```

```
    parser.add_argument(
        "--rejection_reason",
        type=str,
        default=" ",
        help="What's the rejection reason?",
    )
    parser.add_argument(
        "--applied_adjustments",
        type=str,
        default=" ",
        help="What's the applied adjustments?",
    )
    parser.add_argument("--is_example", type=int, default=0, help="What's the example?")
    parser.add_argument(
        "--concerned_people_searched_opinion",
        type=bool,
        default=True,
        help="What's the searched opinion of the concerned people?",
    )
    parser.add_argument(
        "--concerned_people_searched_content",
        type=str,
        default=" ",
        help="What's the searched content of the concerned people?",
    )

    args, _ = parser.parse_known_args()
    return args

```

```
def get_pias(args):

    url = "http://127.0.0.1:3000/pias"
    r = requests.get(url)

    if r.status_code == 200:
        print("The pias are: ")
    else:
        print("There are no pias.")

    pia_ids = json.loads(r.text)
    for pia_id in pia_ids:
        print(pia_id["id"])


def create_pia(args):

    url = "http://127.0.0.1:3000/pias"
    pia = {
        "name": args.name,
        "status": args.status,
        "author_name": args.author_name,
        "evaluator_name": args.evaluator_name,
        "validator_name": args.validator_name,
        "dpo_status": args.dpo_status,
        "dpo_opinion": args.dpo_opinion,
        "dpos_names": args.dpos_names,
        "people_names": args.people_names,
        "concerned_people_opinion": args.concerned_people_opinion,
        "concerned_people_status": args.concerned_people_status,
        "rejection_reason": args.rejection_reason,
        "applied_adjustments": args.applied_adjustments,
        "is_example": args.is_example,
        "concerned_people_searched_opinion": args.concerned_people_searched_opinion,
        "concerned_people_searched_content": args.concerned_people_searched_content,
    }
```

```
    r = requests.post(url, json=pia)

    if r.status_code == 201:
        print("The pia is created!")
        # Edw pera
    else:
        print(f"The pia was not created!")

    pia_id = json.loads(r.text)
    print("pia id:" + str(pia_id["id"]))
    return pia_id


def get_pias_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The pia with pia id: {args.pia_id}, is:", json.loads(r.text))
        return json.loads(r.text)
    else:
        print(f"There is no pia with {args.pia_id} pia id.")


```

```
def patch_pias(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}"
    payload = {
        "name": args.name,
        "status": args.status,
        "author_name": args.author_name,
        "evaluator_name": args.evaluator_name,
        "validator_name": args.validator_name,
        "dpo_status": args.dpo_status,
        "dpo_option": args.dpo_option,
        "dpo_name": args.dpo_name,
        "people_name": args.people_name,
        "concerned_people_opinion": args.concerned_people_opinion,
        "concerned_people_status": args.concerned_people_status,
        "rejection_reason": args.rejection_reason,
        "applied_adjustments": args.applied_adjustments,
        "is_example": args.is_example,
        "concerned_people_searched_opinion": args.concerned_people_searched_opinion,
        "concerned_people_searched_content": args.concerned_people_searched_content,
    }
    r = requests.patch(url, json=payload)

    if r.status_code == 200:
        print("The pia is successfully updated!")
    else:
        print("The update failed.")


```

```
182
183
184  def delete_pias(args):
185
186      url = f"http://127.0.0.1:3000/pias/{args.pia_id}"
187      r = requests.delete(url)
188
189      if r.status_code == 204:
190          print(f"The pia with {args.pia_id} is deleted!")
191      else:
192          print(f"The pia with {args.pia_id} isn't deleted.")
193
194
195  if __name__ == "__main__":
196      pass
197
```

```
1   import argparse
2   import requests, json
3
4
5   def parse_arguments():
6       parser = argparse.ArgumentParser()
7       parser.add_argument(
8           "--pia_id",
9           type=int,
10          default=0,
11          help="Which is the pia_id?",
12      )
13
14      args, _ = parser.parse_known_args()
15      return args
16
17
18  def create_duplicate(args):
19
20      url = "http://127.0.0.1:3000/pias/{args.pia_id}"
21
22      r = requests.get(url)
23      pia = json.loads(r.text)
24
25      r = requests.post(url + "/duplicate", data=pia)
26
27      if r.status_code == 200:
28          print("The pia is duplicated!")
29      else:
30          print(f"The pia was not duplicated!")
31
```

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--pia_id",
        type=int,
        default=0,
        help="Which is the pia_id?",
    )
    parser.add_argument(
        "--reference_to",
        type=str,
        default=" ",
        help="Which is the reference?",
    )
    parser.add_argument(
        "--answer_id",
        type=int,
        default=0,
        help="Which is the answer_id in the pia?",
    )
    parser.add_argument(
        "--data", type=str, default=" ", help="Which are the data of the aanswer?"
    )

    args, _ = parser.parse_known_args()
    my_data = json.loads(args.data)
    args.data = my_data
    return args
```

```python
def get_answers(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/answers"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The answers of the pia with id {args.pia_id}, are:")
    else:
        print("There are no answers for this particular pia")

    answer_ids = json.loads(r.text)
    for answer_id in answer_ids:
        print(answer_id["id"])


def post_answers(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/answers"
    answers = {
        "pia_id": args.pia_id,
        "reference_to": args.reference_to,
        "data": args.data,
    }
    r = requests.post(url, json=answers)

    if r.status_code == 201:
        print(f"The answer with {args.pia_id} is posted!")
    else:
        print("the answer was not posted.")

    answer_id = json.loads(r.text)
    print(f"Answer id: {answer_id['id']}")
```

```
67
68  def get_answers_id(args):
69
70      url = f"http://127.0.0.1:3000/pias/{args.pia_id}/answers/{args.answer_id}"
71      r = requests.get(url)
72
73      if r.status_code == 200:
74          print(f"The answer with {args.answer_id} is:")
75      else:
76          print(f"The answer with the {args.answer_id}, does not exist.")
77
78
79  def patch_answers_id(args):
80
81      url = f"http://127.0.0.1:3000/pias/{args.pia_id}/answers/{args.answer_id}"
82      payload = {"pia_id": args.pia_id, "reference_to": args.regerence_to}
83      r = requests.patch(url, json=payload)
84
85      if r.status_code == 200:
86          print("The answer is updated!")
87      else:
88          print("The answer was not updated")
89
90
91  def delete_answers_id(args):
92
93      url = f"http://127.0.0.1:3000/pias/{args.pia_id}/answers/{args.answer_id}"
94      r = requests.delete(url)
95
96      if r.status_code == 204:
97          print(f"The answer with args.answer_id is deleted.")
98      else:
99          print(f"The answer with {args.answer_id} was not deleted.")
100
```

```
100
101
102 if __name__ == "__main__":
103     pass
104
```

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--pia_id",
        type=int,
        default=0,
        help="Which pia_id",
    )
    parser.add_argument(
        "--description",
        type=str,
        default=" ",
        help="Which is the description of the comment?",
    )
    parser.add_argument(
        "--reference_to",
        type=str,
        default=" ",
        help="Where is the comment referenced to?",
    )
    parser.add_argument(
        "--for_measure",
        type=bool,
        default=True,
        help="Which measure?",
    )

```

```python

def get_comments(args):

    url = "http://127.0.0.1:3000/pias/{args.pia_id}/comments"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The comments of {args.pia_id} are: ")
    else:
        print("There are no comments in this pia!")

    evaluation_ids = json.loads(r.text)
    for evaluation_id in evaluation_ids:
        print(evaluation_id["id"])


def post_comments(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/comments"
    comments = {
        "pia_id": args.pia_id,
        "description": args.description,
        "reference_to": args.reference_to,
        "for_measure": args.for_measure,
    }
    r = requests.post(url, json=comments)

    if r.status_code == 201:
        print("The comment is created!")
    else:
        print("The comment is not created")

```

```
    comment_id = json.loads(r.text)
    print(f"comment id: {comment_id['id']}")


def get_comments_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/comments/{args.comments_id}"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The comment with {args.comments_id} is:")
    else:
        print("This comment id does not exist.")


def patch_comments_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/comments/{args.comments_id}"
    payload = {
        "pia_id": args.pia_id,
        "description": args.description,
        "reference_to": args.reference_to,
        "for_measure": args.for_measure,
    }

    r = requests.patch(url, json=payload)

    if r.status_code == 200:
        print(f"The comment is updated.")
    else:
        print("The comments is not upadated.")


```

```

def delete_comments_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/comments/{args.comments_id}"
    r = requests.delete(url)

    if r.status_code == 204:
        print("The comment is deleted.")
    else:
        print("The comment is not deleted.")


if __name__ == "__main__":
    pass

```

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--pia_id",
        type=int,
        default=0,
        help="Which is the pia_id",
    )
    parser.add_argument(
        "--name",
        type=str,
        default=" ",
        help="What's the name of the object the pia refers to?",
    )
    parser.add_argument(
        "--title", type=str, default=" ", help="What's the title of the measure?"
    )
    parser.add_argument(
        "--content",
        type=str,
        default=" ",
        help="What is the content if the measure?",
    )
    parser.add_argument(
        "--placeholder",
        type=str,
        default=" ",
        help="Who is the place holder of the measure?",
    )
    parser.add_argument(
        "--measure_id",
        type=int,
        default=0,
        help="Which is the measure_id?",
    )

    args, _ = parser.parse_known_args()
    return args

def get_measures(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/measures"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The measures of the pia with pia id: {args.pia_id} are:")
    else:
        print("There are no measures for this pia.")

    measures_ids = json.loads(r.text)
    for measures_id in measures_ids:
        print(measures_id["id"])


```

```python
def post_measures(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/measures"
    payload = {
        "name": args.name,
        "pia_id": args.pia_id,
        "title": args.title,
        "content": args.content,
        "placeholder": args.placeholder,
    }
    r = requests.post(url, json=payload)

    if r.status_code == 201:
        print("The measure is created.")
    else:
        print("The measure is not created.")

    measures_id = json.loads(r.text)
    print(f"Measure id: {measures_id['id']}")


def get_measures_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/measures/{args.measure_id}"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The measure with pia id: {args.pia_id}, is:")
    else:
        print("There is no measure for this pia.")

```

```python

def patch_measures_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/measures/{args.measure_id}"
    payload = {
        "name": args.name,
        "pia_id": args.pia_id,
        "title": args.title,
        "content": args.content,
        "placeholder": args.placeholder,
    }
    r = requests.patch(url, json=payload)

    if r.status_code == 200:
        print("The measure is updated.")
    else:
        print("The measure is not updated.")


def delete_measures_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/measures/{args.measure_id}"
    r = requests.delete(url)

    if r.status_code == 204:
        print("The measure is deleted.")
    else:
        print("The measure is not deleted.")


if __name__ == "__main__":
    pass

```

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--pia_id",
        type=int,
        default=0,
        help="Which is the pia_id?",
    )
    parser.add_argument(
        "--name",
        type=str,
        default=" ",
        help="Which is the name of the evaluation?",
    )
    parser.add_argument(
        "--status", type=int, default=0, help="What is the status of the evaluation?"
    )
    parser.add_argument(
        "--reference_to",
        type=str,
        default=" ",
        help="Where the evaluation is referenced to?",
    )
    parser.add_argument(
        "--action_plan_comment",
        type=str,
        default=" ",
        help="Which is the action plan comment of the evaluation?",
    )
```

```python
    )
    parser.add_argument(
        "--evaluation_comment",
        type=str,
        default=" ",
        help="Which is the evaluation's comment?",
    )
    parser.add_argument(
        "--evaluation_date",
        type=str,
        default=" ",
        help="Which is the evaluation's date?",
    )
    parser.add_argument(
        "--gauges",
        type=str,
        default={},
        help="Which are the gauges of the particular evaluation?",
    )
    parser.add_argument(
        "--estimated_implementation_date",
        type=str,
        default=" ",
        help="Which is the estimated implementation date?",
    )
    parser.add_argument(
        "--person_in_charge", type=str, default=" ", help="Who is the person in charge?"
    )
    parser.add_argument(
        "--global_status",
        type=int,
        default=0,
        help="What the clobal status of the evaluation?",
    )
```

```
    parser.add_argument(
        "--evaluation_id", type=int, default=0, help="Which is the evaluation's id?"
    )

    args, _ = parser.parse_known_args()
    my_gauges = json.loads(args.data)
    args.gauges = my_gauges
    return args

def get_evaluations(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/evaluations"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The evaluations of the pia id: {args.pia_id}, are: ")
    else:
        print("There are no evaluations fot this pia id.")

    evaluation_ids = json.loads(r.text)
    for evaluation_id in evaluation_ids:
        print(evaluation_id["id"])

```

```
def post_evaluations(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/evaluations"
    payload = {
        "name": args.name,
        "reference_to": args.reference_to,
        "pia_id": args.pia_id,
        "status": args.status,
        "action_plan_comment": args.action_plan_comment,
        "evaluation_comment": args.evaluation_comment,
        "evaluation_date": args.evaluation_date,
        "gauges": args.gauges,
        "estimated_implementation_date": args.estimated_implementation_date,
        "person_in_charge": args.person_in_charge,
        "global_status": args.global_status,
    }

    r = requests.post(url, json=payload)

    if r.status_code == 201:
        print("The evaluation is created.")
    else:
        print("The evaluation is not created.")

    evaluation_id = json.loads(r.text)
    print(f"Evaluation id: {evaluation_id['id']}")


```

```python

def get_evaluations_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/evaluations/{args.evaluation_id}"
    r = requests.get(url)

    if r.status_code == 200:
        print(f"The evaluation with the evaluation id: {args.evaluation_id}, is: ")
    else:
        print("There is no evaluation with the particular id.")


def patch_evaluations_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/evaluations/{args.evaluation_id}"
    payload = {
        "name": args.name,
        "reference_to": args.reference_to,
        "pia_id": args.pia_id,
        "status": args.status,
        "action_plan_comment": args.action_plan_comment,
        "evaluation_comment": args.evaluation_comment,
        "evaluation_date": args.evaluation_date,
        # "gauges": {},
        "estimated_implementation_date": args.estimated_implementetion_date,
        "person_in_charge": args.person_in_charge,
        "global_status": args.global_status,
    }
    r = requests.patch(url, json=payload)
    if r.status_code == 200:
        print("The evaluation is updated.")
    else:
        print("The evaluation has not been updated.")


```

```python

def delete_evaluations_id(args):

    url = f"http://127.0.0.1:3000/pias/{args.pia_id}/evaluations/{args.evaluation_id}"
    r = requests.delete(url)

    if r.status_code == 204:
        print("The evaluation is deleted.")
    else:
        print("the evaluation is not deleted.")


if __name__ == "__main__":
    pass

```

```python
import requests, json
import argparse


def parse_arguments():
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "--name",
        type=str,
        default=" ",
        help="What's the name of the structure?",
    )
    parser.add_argument(
        "--sector_name",
        type=int,
        default=0,
        help="What's the sector name of the structure?",
    )
    parser.add_argument(
        "--data",
        type=str,
        default=" ",
        help="Which are the data of the structure?",
    )
    parser.add_argument(
        "--structure_id",
        type=int,
        default=0,
        help="Which is the structure id?",
    )

    args, _ = parser.parse_known_args()
    return args
```

```python

def get_structures(args):

    url = "http://127.0.0.1:3000/structures"
    r = requests.get(url)

    if r.status_code == 200:
        print("The structures are: ")
    else:
        print("There are no structures.")

    structures_ids = json.loads(r.text)
    for structure_id in structures_ids:
        print(structure_id["id"])


def post_structures(args):

    url = "http://127.0.0.1:3000/structures"
    payload = {"name": args.name, "sector_name": args.sector_name, "data": args.data}
    r = requests.post(url, json=payload)

    if r.status_code == 201:
        print("The structure is created.")
    else:
        print("The structure is not created.")


```

```
62
63  def get_structures_id(args):
64
65      url = f"http://127.0.0.1:3000/structures/{args.structure_id}"
66      r = requests.get(url)
67
68      if r.status_code == 200:
69          print(f"The structure with structure id {args.structure_id} is: ")
70      else:
71          print("There is no structure with the specific id.")
72
73      structures_id = json.loads(r.text)
74      print(f"Structure id: {structures_id['id']}")
75
76
77  def patch_structures_id(args):
78
79      url = f"http://127.0.0.1:3000/structures/{args.structure_id}"
80      body = {"name": args.name, "sector_name": args.sector_name, "data": args.data}
81      r = requests.get(url, json=body)
82
83      if r.status_code == 200:
84          print("The structure is updated.")
85      else:
86          print("The structure has not been updated.")
87
```

```
88
89  def delete_structures_id(args):
90
91
92      url = f"http://127.0.0.1:3000/structures/{args.structure_id}"
93      r = requests.delete(url)
94
95      if r.status_code == 204:
96          print("The structure is updated.")
97      else:
98          print("The structure has not been updated.")
99
100
101 if __name__ == "__main__":
102     pass
103
```

```
__dispatch_table = {}


def register_function(name, function):

    if name not in __dispatch_table:
        __dispatch_table[name] = function
        return True

    return False


def run_function(name):

    if name in __dispatch_table:
        return __dispatch_table[name]()

    return None


def print_table():
    print(__dispatch_table)

```

```python
from blockchain import *
import pia_operations
import answer_operations
import comments_operations
import evaluation_operations
import measures_operations
import structures_operations
import get_operation
import dispatch_parsers


get_operation.register_function("create_pia", pia_operations.create_pia)
get_operation.register_function("get_pias_id", pia_operations.get_pias_id)
get_operation.register_function("get_pias", pia_operations.get_pias)
get_operation.register_function("patch_pia", pia_operations.patch_pias)
get_operation.register_function("delete_pias", pia_operations.delete_pias)
dispatch_parsers.register_function("create_pia", pia_operations.parse_arguments)
dispatch_parsers.register_function("get_pias_id", pia_operations.parse_arguments)
dispatch_parsers.register_function("get_pias", pia_operations.parse_arguments)
dispatch_parsers.register_function("patch_pia", pia_operations.parse_arguments)
dispatch_parsers.register_function("delete_pias", pia_operations.parse_arguments)

get_operation.register_function("get_answers", answer_operations.get_answers)
get_operation.register_function("post_answers", answer_operations.post_answers)
get_operation.register_function("get_answers_id", answer_operations.get_answers_id)
get_operation.register_function("patch_answers_id", answer_operations.patch_answers_id)
get_operation.register_function(
    "delete_answers_id", answer_operations.delete_answers_id
)
dispatch_parsers.register_function("get_answers", answer_operations.parse_arguments)
dispatch_parsers.register_function("post_answers", answer_operations.parse_arguments)
dispatch_parsers.register_function("get_answers_id", answer_operations.parse_arguments)
dispatch_parsers.register_function(
    "patch_answers_id", answer_operations.parse_arguments
)
```

```python
dispatch_parsers.register_function(
    "delete_answers_id", answer_operations.parse_arguments
)

get_operation.register_function("get_comments", comments_operations.get_comments)
get_operation.register_function("post_comments", comments_operations.post_comments)
get_operation.register_function("get_comments_id", comments_operations.get_comments_id)
get_operation.register_function(
    "patch_comments_id", comments_operations.patch_comments_id
)
get_operation.register_function(
    "delete_comments_id", comments_operations.delete_comments_id
)
dispatch_parsers.register_function("get_comments", comments_operations.parse_arguments)
dispatch_parsers.register_function("post_comments", comments_operations.parse_arguments)
dispatch_parsers.register_function(
    "get_comments_id", comments_operations.parse_arguments
)
dispatch_parsers.register_function(
    "patch_comments_id", comments_operations.parse_arguments
)
dispatch_parsers.register_function(
    "delete_comments_id", comments_operations.parse_arguments
)

get_operation.register_function(
    "get_evaluations", evaluation_operations.get_evaluations
)
get_operation.register_function(
    "post_evaluations", evaluation_operations.post_evaluations
)
get_operation.register_function(
    "get_evaluations_id", evaluation_operations.get_evaluations_id
)
```

```
get_operation.register_function(
    "patch_evaluations_id", evaluation_operations.patch_evaluations_id
)
get_operation.register_function(
    "delete_evaluations_id", evaluation_operations.delete_evaluations_id
)
dispatch_parsers.register_function(
    "get_evaluations", evaluation_operations.parse_arguments
)
dispatch_parsers.register_function(
    "post_evaluations", evaluation_operations.parse_arguments
)
dispatch_parsers.register_function(
    "get_evaluations_id", evaluation_operations.parse_arguments
)
dispatch_parsers.register_function(
    "patch_evaluations_id", evaluation_operations.parse_arguments
)
dispatch_parsers.register_function(
    "delete_evaluations_id", evaluation_operations.parse_arguments
)

get_operation.register_function("get_measures", measures_operations.get_measures)
get_operation.register_function("post_measures", measures_operations.post_measures)
get_operation.register_function("get_measure_id", measures_operations.get_measures_id)
get_operation.register_function(
    "patch_measures_id", measures_operations.patch_measures_id
)
get_operation.register_function(
    "delete_measures_id", measures_operations.delete_measures_id
)
dispatch_parsers.register_function("get_measures", measures_operations.parse_arguments)
dispatch_parsers.register_function("post_measures", measures_operations.parse_arguments)
dispatch_parsers.register_function(
    "get_measures_id", measures_operations.parse_arguments
)
```

```
dispatch_parsers.register_function(
    "patch_measures_id", measures_operations.parse_arguments
)
dispatch_parsers.register_function(
    "delete_measures_id", measures_operations.parse_arguments
)

get_operation.register_function("get_structures", structures_operations.get_structures)
get_operation.register_function(
    "post_structures", structures_operations.post_structures
)
get_operation.register_function(
    "get_structures_id", structures_operations.get_structures_id
)
get_operation.register_function(
    "patch_structures_id", structures_operations.patch_structures_id
)
get_operation.register_function(
    "delete_structures_id", structures_operations.delete_structures_id
)
dispatch_parsers.register_function(
    "get_structures", structures_operations.parse_arguments
)
dispatch_parsers.register_function(
    "post_structures", structures_operations.parse_arguments
)
dispatch_parsers.register_function(
    "get_structures_id", structures_operations.parse_arguments
)
dispatch_parsers.register_function(
    "patch_structures_id", structures_operations.parse_arguments
)
```

```
dispatch_parsers.register_function(
    "delete_structures_id", structures_operations.parse_arguments
)

args = get_operation.parse_arguments()
get_operation.run_function(
    args.operation, dispatch_parsers.run_function(args.operation)
)

```

Appendix D - Παρουσίαση

```python
import argparse
from os import listdir
from os.path import isfile, join
from blockchain import *

__dispatch_table = {}


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--operation",
        type=str,
        default=None,
        help="Specify operation for the system to execute",
    )

    args, _ = parser.parse_known_args()
    return args


def register_function(name, function):

    if name not in __dispatch_table:
        __dispatch_table[name] = function
        return True

    return False


def existed_pias(name, args):
    old_pia = __dispatch_table[name](args)
    return old_pia

```

```python

def run_function(name, args):

    if name in __dispatch_table:

        if not "get" in name:

            #pia = __dispatch_table[name](args)

            with open("html/index.html", "r") as template_file:
                read_template = template_file.read()

            cwd = "/home/aris/thesis/"
            onlyfiles = [f for f in listdir(cwd) if isfile(join(cwd, f))]

            max_file_id = 0
            for filename in onlyfiles:
                if "file" in filename:
                    tokens = filename.split("file")
                    get_file_number = tokens[1].split(".")[0]
                    max_file_id = max(max_file_id, int(get_file_number))

```

```
            pia = __dispatch_table[name](args)
            template_mappings = {
                "Name1": pia["name"],
                "Status1": pia["status"],
                "Author1": pia["author_name"],
                "Evaluator1": pia["evaluator_name"],
                "Validator1": pia["validator_name"],
                "DPO1": pia["dpo_status"],
                "DPO2": pia["dpo_opinion"],
                "DPO3": pia["dpos_names"],
                "People1": pia["people_names"],
                "Concerned1": pia["concerned_people_opinion"],
                "Concerned2": pia["concerned_people_status"],
                "Rejection1": pia["rejection_reason"],
                "Applied1": pia["applied_adjustments"],
                "Example1": pia["is_example"],
                "Concerned3": pia["concerned_people_searched_opinion"],
                "Concerned4": pia["concerned_people_searched_content"],
                "OBC" : max_file_id
            }

            for placeholder, value in template_mappings.items():
                read_template = read_template.replace(placeholder, str(value))

            with open("html/index3.html", "w") as file_with_data:
                file_with_data.write(read_template)

```

```
            blockchain = Blockchain()


            # print(f"The maximum file id is {max_file_id}")

            new = blockchain.new_transaction(
                "terezaki_aristea", "markakis_evaggelos", pia
            )
            last_file_id = max_file_id + 1
            blockchain.new_block()
            new_path = f"file{last_file_id}.txt"
            blockchain.persist(new_path)
            print( f"A new block has been added. Overall block count = {last_file_id}")

            # blockchain.unpersist(new_path)
            return True
        else:
            with open("html/index.html", "r") as template_file:
                read_template = template_file.read()

            cwd = "/home/aris/thesis/"
            onlyfiles = [f for f in listdir(cwd) if isfile(join(cwd, f))]

            max_file_id = 0
            for filename in onlyfiles:
                if "file" in filename:
                    tokens = filename.split("file")
                    get_file_number = tokens[1].split(".")[0]
                    max_file_id = max(max_file_id, int(get_file_number))

```

```
            pia = __dispatch_table[name](args)
            template_mappings = {
                "Name1": pia["name"],
                "Status1": pia["status"],
                "Author1": pia["author_name"],
                "Evaluator1": pia["evaluator_name"],
                "Validator1": pia["validator_name"],
                "DPO1": pia["dpo_status"],
                "DPO2": pia["dpo_opinion"],
                "DPO3": pia["dpos_names"],
                "People1": pia["people_names"],
                "Concerned1": pia["concerned_people_opinion"],
                "Concerned2": pia["concerned_people_status"],
                "Rejection1": pia["rejection_reason"],
                "Applied1": pia["applied_adjustments"],
                "Example1": pia["is_example"],
                "Concerned3": pia["concerned_people_searched_opinion"],
                "Concerned4": pia["concerned_people_searched_content"],
                "OBC" : max_file_id
            }

            for placeholder, value in template_mappings.items():
                read_template = read_template.replace(placeholder, str(value))

            with open("html/index2.html", "w") as file_with_data:
                file_with_data.write(read_template)

            return True

    return False
```

```
<html>
    <head>
        <title>
            Pia Blockchain
        </title>
        <link rel="stylesheet" type= "text/css" href="stylesheet.css"/>
    </head>
    <body>
        <img src="Blockchain.jpg" class="center"
        <div class= "filter">
        </div>
        <table>

        <tr><th colspan= "2"><h1 align= center> Results of the PIA Block</h1></th></tr>
                    <tr><td><h4>Name</h4></td>
                        <td>Name1 </td></tr>
                    <tr><td><h4>Status</td>
                        <td> Status1 </td></tr>
                    <tr><td><h4>Author Name</td>
                        <td> Author1</td></tr>
                    <tr><td><h4>Evaluator Name</td>
                        <td> Evaluator1</td></tr>
                    <tr><td><h4>Validator Name</td>
                        <td> Validator1 </td></tr>
                    <tr><td><h4>DPO Status</td>
                        <td> DPO1</td></tr>
                    <tr><td><h4>DPO Opinion</td>
                        <td> DPO2</td></tr>
                    <tr><td><h4>DPO Name</td>
                        <td> DPO3</td></tr>
                    <tr><td><h4>People Name</td>
                        <td> People1</td></tr>
```

```
32                      <td> People1</td></tr>
33                  <tr><td><h4>Concerned People Opinion</td>
34                      <td> Concerned1</td></tr>
35                  <tr><td><h4>Concerned People Status</td>
36                      <td> Concerned2</td></tr>
37                  <tr><td><h4>Rejection Reason</td>
38                      <td> Rejection1</td></tr>
39                  <tr><td><h4>Applied Adjustments</td>
40                      <td>Applied1</td></tr>
41                  <tr><td><h4>Is Example</td>
42                      <td> Example1</td></tr>
43                  <tr><td><h4>Concerned People Searched Opinion</td>
44                      <td> Concerned3</td></tr>
45                  <tr><td><h4>Concerned People Searched Content</td>
46                      <td> Concerned4</td></tr>
47                  <tr><td><h4>Overall Block count</h4></td>
48                      <td>OBC</td></tr>
49          </table>
50      </body>
51  </html>
52
```