ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός – δημιουργία λογισμικού διαχείρισης αρχείων με την γλώσσα python

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Του

ΘΕΟΔΩΡΙΔΗ ΓΡΗΓΟΡΙΟΥ

(ΑΕΜ: 2216)

Και του

ΠΑΠΑΝΙΚΟΥ ΠΑΓΙΩΤΗ

(ΑΕΜ: 1997)

Επιβλέπων: **Δόσης Μιχαήλ**

Καθηγητής

Καστοριά Απρίλιος - 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός – δημιουργία λογισμικού διαχείρισης αρχείων με την γλώσσα python

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Του

ΘΕΟΔΩΡΙΔΗ ΓΡΗΓΟΡΙΟΥ

(ΑΕΜ: 2216)

Και του

ΠΑΠΑΝΙΚΟΥ ΠΑΓΙΩΤΗ

(ΑΕΜ: 1997)

Επιβλέπων: **Δόσης Μιχαήλ**
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26/04/2023

.....	.....	.....
Δόσης Μιχαήλ	Δημόκας Νικόλαος	Νικολάου Σπυρίδων.

Καστοριά Απρίλιος - 2023

Περίληψη

Από την πρώτη εμφάνιση των λειτουργικών συστημάτων σε παραθυρικό περιβάλλον εμφανίστηκαν και οι πρώτοι διαχειριστές αρχείων, οι οποίοι βρίσκονται προ εγκατεστημένοι στο λειτουργικό, δημιουργώντας ένα φιλικό και πλήρως κατανοητό περιβάλλον για το μέσο χρήστη . Ωστόσο, με το πέρασμα του χρόνου και την εξέλιξη της τεχνολογίας ,αυτοί οι προεγκατεστημένοι διαχειριστές αρχείων σταμάτησαν να καλύπτουν τις ανάγκες μερικών πιο απαιτητικών χρηστών με αποτέλεσμα να δημιουργηθούν εφαρμογές τρίτων για την κάλυψη των αναγκών τους. Αυτές οι εφαρμογές ξεχωρίζουν για την πληθώρα επιλογών παραμετροποίησης και ελευθερίας που προσφέρουν στο χρήστη, καθώς και τη δυνατότητα να αποθηκεύουν αρχεία στο διαδίκτυο, δίνοντας την ευκαιρία για πρόσβαση από οποιονδήποτε και οπουδήποτε. Στόχοι για ανάπτυξη της εφαρμογής είναι χρήση άλλων τεχνολογιών και μεθόδων για την επίτευξη μοντέρνου σχεδιασμού φιλικό και ευπαρουσίαστο προς τον χρήστη, δημιουργία δωματίων τηλεδιάσκεψης μεταξύ των χρηστών της εφαρμογής, δημιουργίας και ανταλλαγής σημειώσεων , δυνατότητα ανεβάσματος αρχείου με προστασία κωδικού, πρόσβαση αρχείων με βάση των δικαιωμάτων του χρήστη.Κύριο προτέρημα της εφαρμογής CRUD σε σχέση με τις υφιστάμενες εφαρμογές είναι ότι η βάση δεδομένων μπορεί να εγκατασταθεί σε οποιονδήποτε server που επιθυμεί η εταρεία που την χρησιμοποιεί.

Λέξεις Κλειδιά: Python, μίας χρήσης κωδικός πρόσβασης, Xampp, OTP, Λογισμικό διαχείρισης αρχείων, Βάση δεδομένων, χρήστες.

Abstract

From the first appearance of operating systems in a windowed environment, the first file managers appeared, which are pre-installed in the operating system, creating a friendly and fully understandable environment for the average user. However, over time and with the evolution of technology, these pre-installed file managers stopped meeting the needs of some more demanding users, resulting in the creation of third-party applications to meet their needs. These applications stand out for the plethora of customization options and freedom they offer to the user, as well as the ability to store files on the internet, providing the opportunity for access by anyone, anywhere. Goals for the development of the application include the use of other technologies and methods to achieve modern user-friendly and responsive design, the creation of video conference rooms among application users, the creation and sharing of notes, the ability to upload files with password protection, and access to files based on user rights. The main advantage of the CRUD application over existing applications is that the database can be installed on any server desired by the company using it.

Key words: Python, one time password, Xampp, OTP, File management system, database, users.

Περιεχόμενα

Λίστα εικόνων

Εισαγωγή

Σκοπός της εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η δημιουργία μιας εφαρμογής Python για τη διαχείριση αρχείων.

Για να λειτουργήσει κατάλληλα μια τέτοια εφαρμογή προϋποθέτει τη δημιουργία μια βάσης δεδομένων έτσι ώστε να γίνεται διαχείριση και αποθήκευση των δεδομένων από ένα σύστημα διαχείρισης βάσης δεδομένων. Για τη συγκεκριμένη εφαρμογή έγινε η χρήση της MySQL μέσω του phpMyAdmin.

Στην εφαρμογή γίνεται η υλοποίηση της διαχείρισης των χρηστών και των αρχείων. Ο χρήστης έχει την δυνατότητα να ανεβάσει είτε να κατεβάσει αρχεία ενώ ο διαχειριστής έχει επιπλέον δικαιώματα για να επεξεργαστεί να διαγράψει ή να ενημερώσει τα ανεβασμένα αρχεία καθώς και τα στοιχεία των χρηστών.

Αρχικά αναφέρεται η χρήση των λογισμικών διαχείρισης και η αναγκαιότητά τους, καθώς και ο τρόπος λειτουργείας τους, δείχνοντας κάποια παραδείγματα παρόμοιων εφαρμογών.

Στη συνέχεια αναλύουμε τον σχεδιασμό της εφαρμογής με τη χρήση διαγραμμάτων UML και συσχετίσεων που είναι απαραίτητα για τη δημιουργία της εφαρμογής. Έπειτα γίνεται η ανάλυση και η δομή της βάσης δεδομένων.

Τέλος αναφέρεται η ανάπτυξη της λειτουργίας της εφαρμογής, η εγκατάστασή της στους ηλεκτρονικούς υπολογιστές καθώς και η αναλυτική παρουσίαση της λειτουργίας και των παραμέτρων της.

Κεφάλαιο 1ο: Λογισμικό διαχείρισης αρχείων και εργαλεία δημιουργίας του.

Ο τομέας της πληροφορικής και της τεχνολογίας όλο ένα και επεκτείνεται με σκοπό την απλοποίηση των εργασιών και ως κύριο στόχο την ελάφρυνση του βάρους και του χρόνου που απαιτείται για να εκτελεστεί μια εργασία από τον άνθρωπο. Έτσι και στην δική μας περίπτωση, θα αναφερθούμε στα λογισμικά διαχείρισης αρχείων. Τα προγράμματα αυτά χρησιμοποιούνται από επιχειρήσεις καθώς και από ομάδες ατόμων και μερικά από τα βασικά τους προτερήματα είναι:

- **Ευκολία:** Χρησιμοποιώντας ένα λογισμικό διαχείρισης αρχείων ο εκάστοτε χρήστης έχει προσβασιμότητα στις πληροφορίες ή τα αρχεία οποιαδήποτε στιγμή επιθυμεί καθώς δουλεύει σε κάποια συγκεκριμένη ανάθεση. Έτσι επιτυγχάνεται η μείωση του χρόνου ολοκλήρωσης της εργασίας και η καταβολή ενέργειας του χρήστη σε αυτή, αυτό οδηγεί στην καλύτερη συγκέντρωση, απόδοση και παραγωγικότητα της ομάδας.
- **Μείωση του κόστους:** Εκτός από τα λειτουργικά έξοδα του οργανισμού, ένας οργανισμός χρειάζεται να επενδύσει χρήματα και πόρους για αποθήκευση, οργάνωση ,διαμοιρασμό και συντήρηση στα αρχεία του. Ωστόσο χρησιμοποιώντας το κατάλληλο λογισμικό διαχείρισης αρχείων μπορεί να γλιτώσει σημαντικό ποσό χρημάτων.
- **Μείωση του χρόνου:** Όπως αναφέραμε παραπάνω το να έχει ένας οργανισμός κοινόχρηστο τρόπο οργάνωσης και διαμοιρασμού αρχείων όχι μόνο γλιτώνει χρήματα αλλά και χρόνο. Αυτό εξασφαλίζεται από την γρήγορη αναζήτηση ανάκτηση των αρχείων με την βοήθεια λογισμικού.
- **Ακεραιότητα δεδομένων:** Η διατήρηση της ασφάλειας των πληροφοριών είναι μια πραγματική πρόκληση. Αλλά χρησιμοποιώντας ένα λογισμικό διαχείρισης αρχείων διασφαλίζονται παράλληλα οι απαραίτητες προφυλάξεις ασφάλειας και την ακεραιότητα των πληροφοριών.

1.1.Παρόμοια προγράμματα διαχείρισης αρχείων.

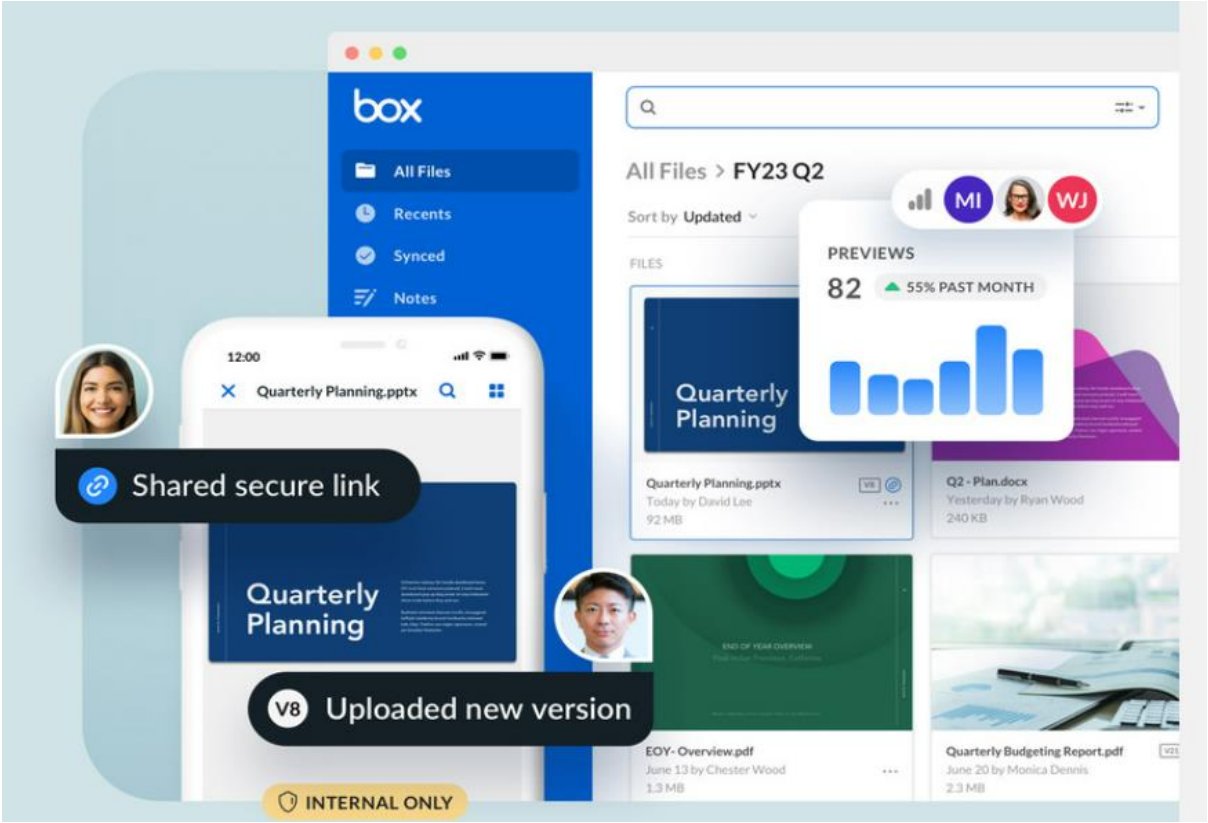
Η διαχείριση αρχείων είναι η διαδικασία οργάνωσης, ενοποίησης και συλλογής πληροφοριών στις διάφορες μορφές της, όπως έγγραφα, αρχεία πολυμέσων και αρχεία σχεδίασης. Σχεδόν οτιδήποτε μπορείτε να αποθηκευτεί στον υπολογιστή θεωρείται αρχείο. Η διαδικασία διαχείρισης τους πραγματοποιείται μέσω μιας ποικιλίας προγραμμάτων και συμπεριλαμβανομένων συστημάτων διαχείρισης αρχείων. Οι πλατφόρμες διαχείρισης εταιρικού περιεχομένου είναι ιδιαίτερα σημαντικές στις μέρες μας καθώς έχει αυξηθεί σημαντικά το ποσοστό των εργαζομένων που δουλεύουν από το σπίτι. Έτσι μπορεί ο κάθε εργαζόμενος να έχει πρόσβαση στα αρχεία που του είναι απαραίτητα να ανεβάζει και να μοιράζεται τη δουλειά του καθώς και να επικοινωνεί με τους συναδέλφους του, κάτι που έχει ως αποτέλεσμα τη καλύτερη ροή των εργατιών της επιχείρησης (Wikipedia, 2023).

1.1.1.BOX

Εικόνα 1[ΚΕΦ.1.]Λογότυπο της εφαρμογής BOX

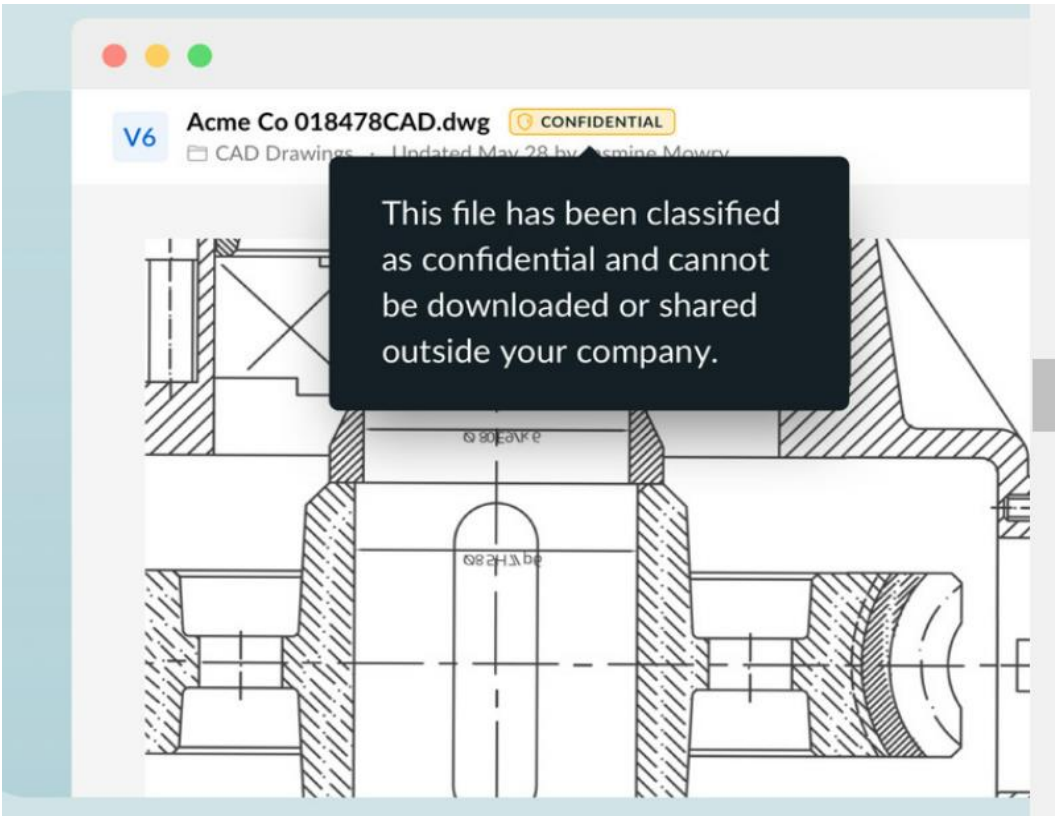
Το BOX είναι ένα σύστημα διαχείρισης αρχείων cloud που κατά κύριο λόγο απευθύνεται σε εταιρίες. Το BOX ξεκίνησε το 2003 με τον Aaron Levie , ο οποίος ήταν φοιτητής επιχειρήσεων στο Πανεπιστήμιο της Νότιας Καλιφόρνια γράφοντας μία εργασία σχετική με την αποθήκευση των ψηφιακών αρχείων στο διαδίκτυο. Το 2005 ο Levie παράτησε το σχολείο και μαζί με τον Dylan Smith ίδρυσαν το BOX.Τ ο Box αρχικά επικεντρώθηκε στους καταναλωτές, αλλά

πολλοί από αυτούς τους καταναλωτές χρησιμοποιούσαν την υπηρεσία στην εργασία τους έτσι το 2009 επικεντρώθηκε στην ενσωμάτωση επιχειρηματικών εφαρμογών. Το BOX επεκτάθηκε διεθνώς και το 2020 ανακοίνωσε μια νέα βελτιωμένη έκδοση που ενσωματώνει λογισμικό βιντεοτηλεφωνίας λόγω της αύξησης απομακρυσμένης εργασίας κατά τη διάρκεια της πανδημίας (Box, 2023).



Εικόνα 2[ΚΕΦ.1.]Περιβάλλον διαχείρισης της εφαρμογής BOX

Το Box προσφέρει προηγμένη κοινή χρήση αρχείων δίνοντας στις επιχειρήσεις έναν ευκολότερο τρόπο κοινής χρήσης και ελέγχου του περιεχόμενου με ασφάλεια μεταξύ προμηθευτών, πελατών και συναδέλφων. Καθιστά εύκολη την διαχείριση, την οργάνωση και τον διαμοιρασμό των αρχείων καθώς και των δικαιωμάτων τους με το φιλικό περιβάλλον διαχείρισης του χρήστη.



Εικόνα 3[ΚΕΦ.1.]Ανεβασμένο αρχείο στην εφαρμογή BOX

1.1.2. Tresorit

tresorit

Εικόνα 4[ΚΕΦ.1.] Λογότυπο της εφαρμογής Tresorit

Το Tresorit είναι ένα ασφαλές εργαλείο αποθήκευσης και συνεργασίας, σχεδιασμένο για επιχειρήσεις και άτομα που εκτιμούν την ιδιωτικότητα και την ασφάλεια των δεδομένων τους. Εδώ είναι μερικά από τα χαρακτηριστικά και τις δυνατότητες που προσφέρει το Tresorit:

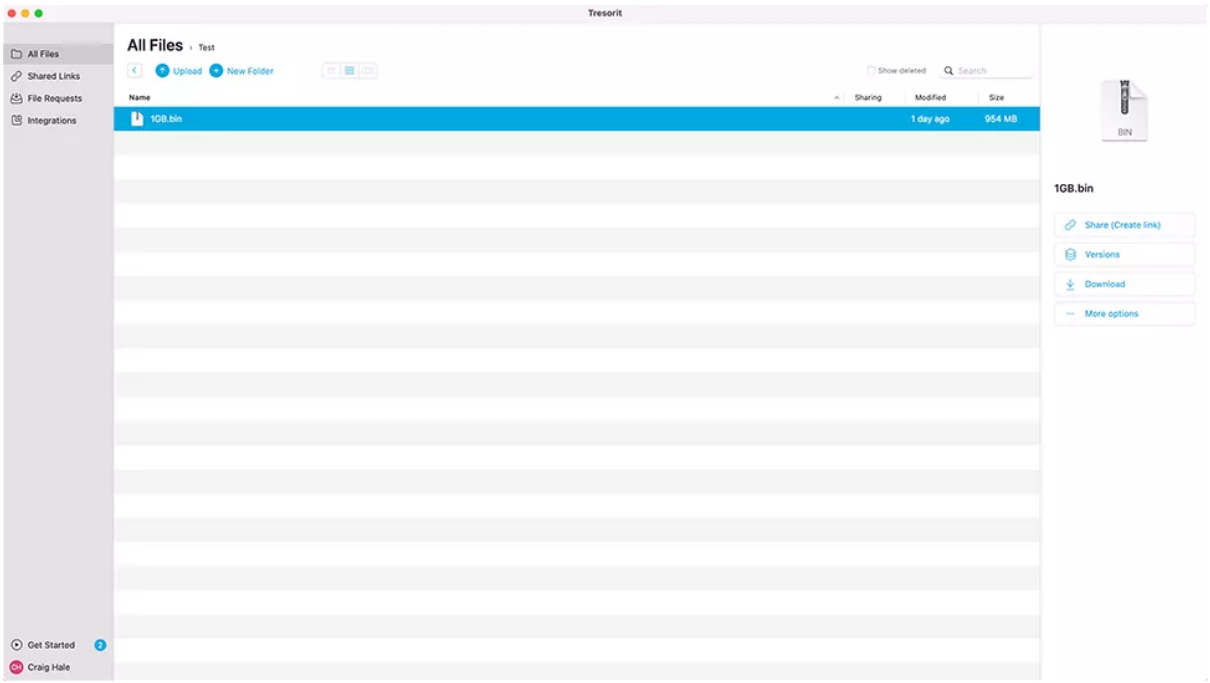
- Κρυπτογράφηση end-to-end: Το Tresorit κρυπτογραφεί τα αρχεία πριν αποχωρήσουν από τη συσκευή και παραμένουν κρυπτογραφημένα κατά τη μεταφορά και στην αποθήκευση. Αυτό εξασφαλίζει ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση στα δεδομένα σας.

- Κρυπτογράφηση zero-knowledge: Το Tresorit χρησιμοποιεί κρυπτογράφηση zero-knowledge, που σημαίνει ότι μόνο ο χρήστης έχει πρόσβαση στα κλειδιά κρυπτογράφησης. Αυτό εξασφαλίζει ότι το Tresorit δεν μπορεί να διαβάσει ή να έχει πρόσβαση στα αρχεία.

- Απομακρυσμένη διαγραφή : Σε περίπτωση κλοπής υπάρχει η δυνατότητα διαγραφής όλων των αρχείων έτσι ώστε να αποτραπεί η μη εξουσιοδοτημένη πρόσβαση των αρχείων.

- Ανάκτηση αρχείων : Η εφαρμογή παρακολουθεί και αποθηκεύει την τροποποίηση των αρχείων αυτό έχει ως σκοπό την επιλογή ανάκτησης προηγούμενου περιεχομένου του αρχείου.

- Φιλικό περιβάλλον διεπαφής του χρήστη: Το Tresorit κάνει εύκολη την πρόσβαση και την διαχείριση αρχείων ακόμα και στους χρήστες που δεν έχουν καμία εμπειρία με παρόμοιο λογισμικό (Tresorit, 2023).



Εικόνα 5 [ΚΕΦ.1.]Περιβάλλον διαχείρισης της εφαρμογής Tresorit

1.1.3. Citrix ShareFile

Εικόνα 6[ΚΕΦ.1.]Λογότυπο της εφαρμογής Citrix ShareFile

Το Citrix ShareFile είναι μια ασφαλής πλατφόρμα σχεδιασμένη κυρίως για επιχειρήσεις ανεξαρτήτως μεγέθους με στόχο τον διαμοιρασμό πληροφοριών μεταξύ των υπαλλήλων τους. Μερικά από τα προτερήματά του που το κάνουν να ξεχωρίζει είναι:

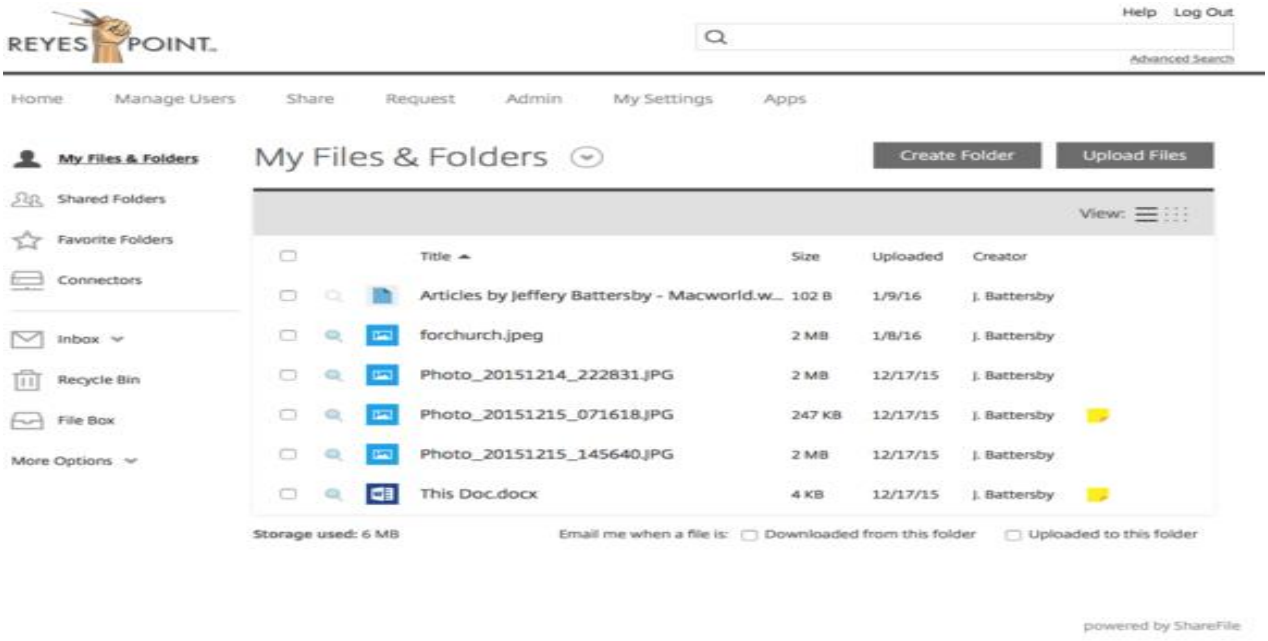
- Ασφαλής μεταφορά αρχείων: Με το Citrix ShareFile οι χρήστες μπορούν να μοιράσουν μεταξύ τους αρχεία εντός ή εκτός της εταιρείας. Τα αρχεία έχουν την δυνατότητα προστασίας με τη χρήση κωδικού πρόσβασης ή να έχουν ημερομηνία λήξης.

- Προσαρμοσμένη επωνυμία και εξατομίκευση: Οι επιχειρήσεις μπορούν να προσαρμόσουν την πλατφόρμα με τη δική τους επωνυμία, λογότυπο και χρώματος της επιλογής τους , δίνοντας μια εξατομικευμένη πινελιά στην εμπειρία κοινής χρήσης αρχείων.

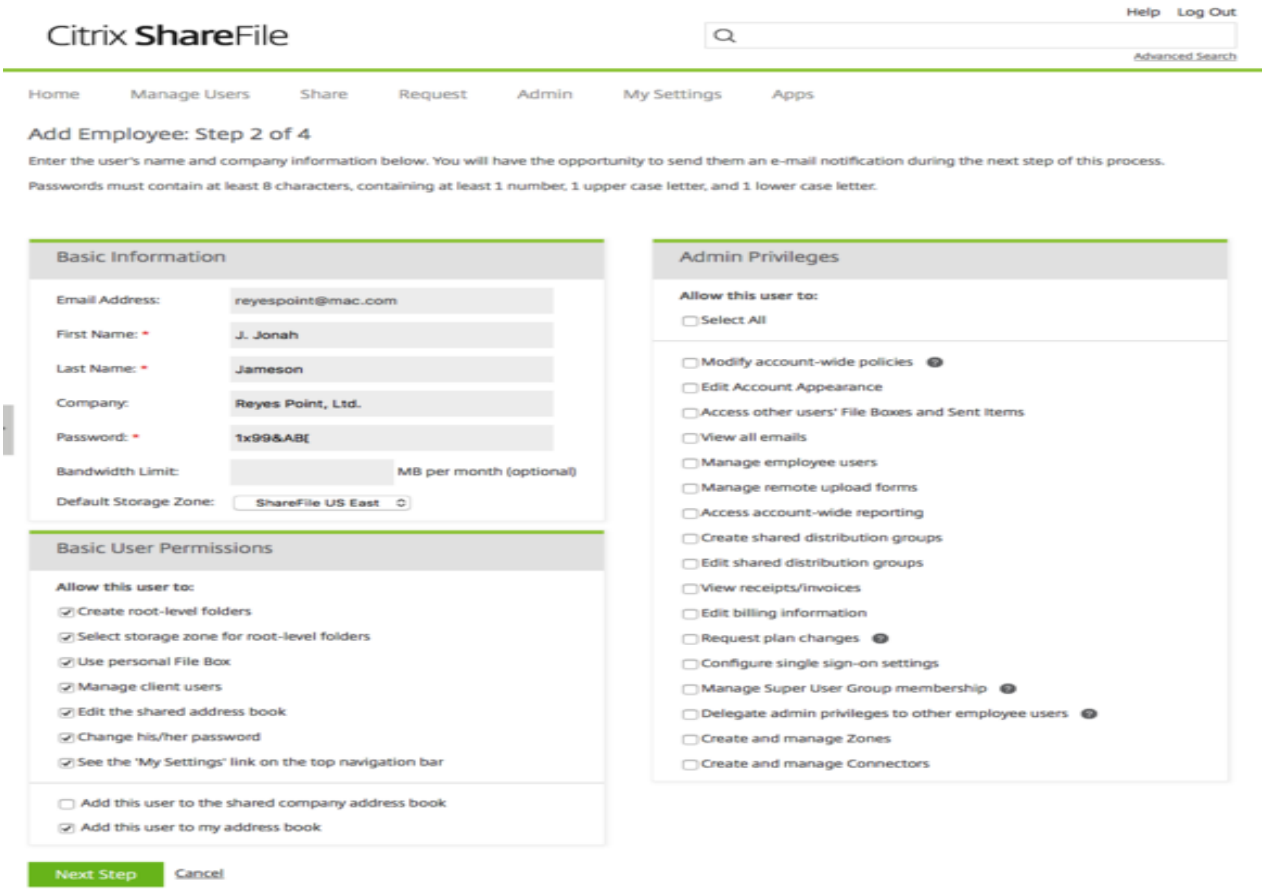
- Προηγμένες δυνατότητες συνεργασίας: Το Citrix ShareFile επιτρέπει σε πολλούς χρήστες να συνεργάζονται και να επεξεργάζονται αρχεία σε πραγματικό χρόνο, με λειτουργίες όπως σχολιασμός εγγράφων, σχόλια και έλεγχος έκδοσης. Οι χρήστες μπορούν επίσης να ρυθμίσουν ροές εργασίας και να αυτοματοποιήσουν τις διαδικασίες για τον εξ ορθολογισμό της συνεργασίας και την αύξηση της παραγωγικότητας

- Ασφαλής συγχρονισμός και πρόσβαση αρχείων: Το Citrix ShareFile παρέχει ασφαλή συγχρονισμό αρχείων σε πολλές συσκευές, διασφαλίζοντας ότι οι χρήστες έχουν πάντα πρόσβαση στα πιο ενημερωμένα αρχεία από οπουδήποτε.

Συνολικά, το Citrix ShareFile είναι μια ολοκληρωμένη και ασφαλής λύση κοινής χρήσης αρχείων και συνεργασίας που μπορεί να βοηθήσει τις επιχειρήσεις να εξορθολογήσουν τις ροές εργασίας τους και να αυξήσουν την παραγωγικότητά τους, διασφαλίζοντας παράλληλα την ασφάλεια και το απόρρητο των δεδομένων τους (Citrix ShareFile, 2023).



Εικόνα 7 [ΚΕΦ.1.] Παράθυρο διαχείρισης χρήστη της εφαρμογής Citrix ShareFile



Εικόνα 8 [ΚΕΦ.1.] Παράθυρο διαχειριστή της εφαρμογής Citrix ShareFIle

1.1.4. Συγκριτική επισκοπήση πλεονεκτημάτων – μειονεκτημάτων σε σχέση με υφιστάμενων εμπορικών εφαρμογών διαχείρισης.

Πλεονεκτήματα:

1. Η εφαρμογή CRUD είναι λειτουργική και μπορεί να χρησιμοποιηθεί χωρίς σύνδεση στο διαδίκτυο, γεγονός που την καθιστά πιο αξιόπιστη και ευέλικτη στη χρήση.
2. Η εφαρμογή μπορεί να εγκατασταθεί σε οποιονδήποτε server επιθυμεί η εταιρεία που τη χρησιμοποιεί, προσφέροντας έτσι ανεξαρτησία και επιλογή στην εταιρεία.

Μειονεκτήματα:

1. Ο μοντέρνος σχεδιασμός της εφαρμογής απουσιάζει, οπότε η εφαρμογή μπορεί να μην είναι τόσο ελκυστική για τους χρήστες.
2. Η κρυπτογράφηση αρχείων λείπει, κάτι που μπορεί να απειλήσει την ασφάλεια των δεδομένων που αποθηκεύονται στην εφαρμογή.
3. Δεν περιλαμβάνεται λειτουργία ανταλλαγής σημειώσεων, η οποία θα μπορούσε να βελτιώσει τη συνεργασία μεταξύ των χρηστών της εφαρμογής.

1.2 Εργαλεία δημιουργίας του λογισμικού διαχείρισης αρχείων με τη γλώσσα προγραμματισμού python

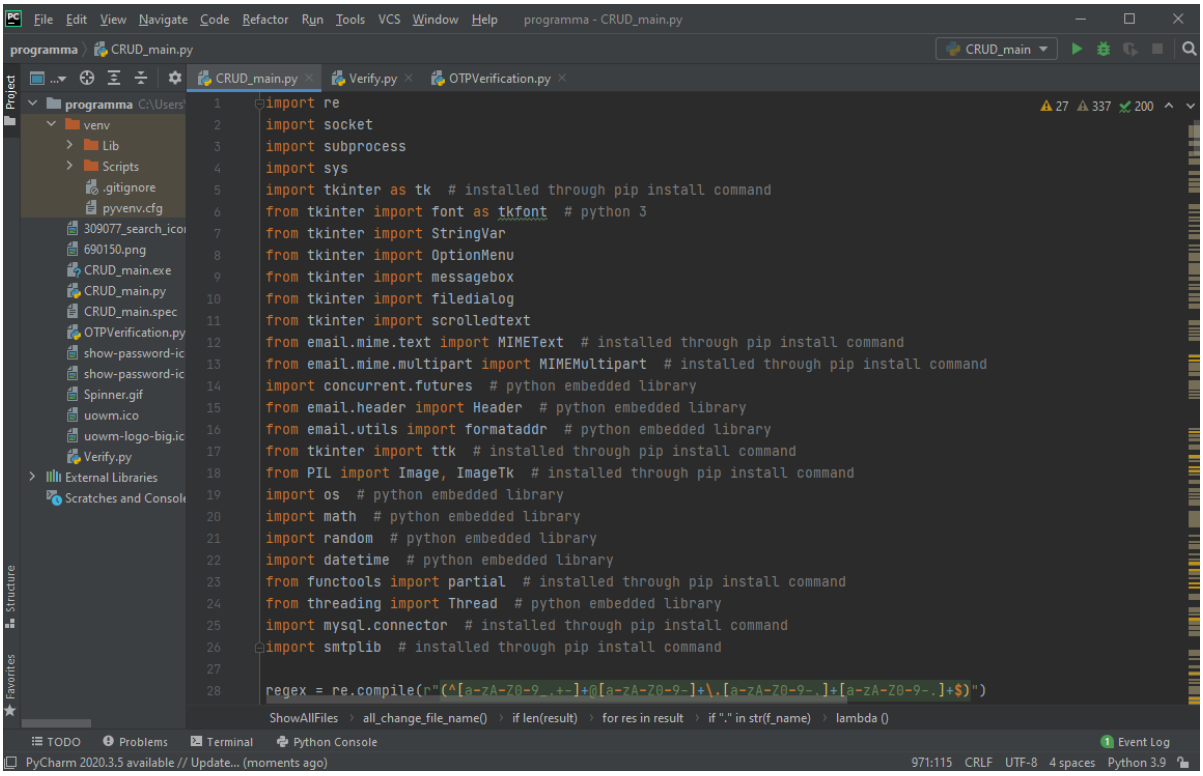
Αρχικά στο σημείου αυτό θα κάνουμε αναφορά στη γλώσσα προγραμματισμού python ,τις δυνατότητές της και τα πλεονεκτήματα που μας παρέχει σε σύγκριση με άλλες γλώσσες προγραμματισμού. Η python είναι διερμηνευόμενη (interpreted), γενικού σκοπού και υψηλού επιπέδου γλώσσα προγραμματισμού. Δημιουργήθηκε από τον Ολλανδό Γκίντο βαν Ρόσσουμ(Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε το 1991.Η γλώσσα αυτή αναπτύσσεται ως ανοιχτό λογισμικό(open source), ο κώδικας διανέμεται με την άδεια Software Foundation License. Τα πλεονεκτήματά της είναι η ευκολία ανάγνωσης και διατύπωσης του κώδικα , καθώς το συντακτικό της επιτρέπει στους προγραμματιστές της εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι σε άλλες γλώσσες προγραμματισμού όπως η C++ ή η java , αυτό όμως γίνεται θυσιάζοντας μέρος της ταχύτητας των προς εκτέλεση προγραμμάτων. Αυτό

επιτυγχάνεται χάρη στις πολλές βιβλιοθήκες που περιλαμβάνουν από ασύγχρονη επεξεργασία έως συμπιεσμένα αρχεία, οι ευκολίες που παρέχει είναι σημαντικές καθώς καλύπτει ένα ευρύ φάσμα πιθανών προβλημάτων όπου αυτό την καθιστά κατάλληλη για αρχάριους προγραμματιστές. Διαθέτει αυτόματη διαχείριση μνήμης όπου αυτό σημαίνει ότι δεν χρειάζεται να ελευθερώνεται η μνήμη από τον εκάστοτε προγραμματιστή που δεσμεύεται δημιουργώντας αντικείμενα ,επίσης η python αντιλαμβάνεται πότε το ίδιο αντικείμενο αναφέρεται πάνω από μία φορές έτσι δεν το αποθηκεύει στην μνήμη εφόσον δεν χρειάζεται. Η τεχνική αυτή ονομάζεται μέτρηση αναφορών(reference counting). Τα προγράμματα τα οποία είναι γραμμένα σε python μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους, του επεξεργαστή αλλά και του λειτουργικού συστήματος(Windows,Linux,MacOs) τα προγράμματα εκτελούνται με τον ίδιο τρόπο (Python, 2023).

1.2.1. Έκδοση Python, δημιουργία του πηγαίου κώδικα και ανάλυση αυτού με το Pycharm IDE.

Για το λογισμικό που έχουμε δημιουργήσει χρησιμοποιήθηκε το πρόγραμμα δημιουργίας προγραμματιστικού κώδικα Pycharm. Το Pycharm είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης εφαρμογών για τη γλώσσα προγραμματισμού Python. Το συγκεκριμένο λογισμικό επιτρέπει την δημιουργία εφαρμογών από ένα σύνολο αρθρωτών περιεχομένων λογισμικού(Modular Software Components),γνωστά και ως Modules.Το Pycharm μπορεί να εκτελεστεί σε λογισμικό περιβάλλον Windows,macOS,Linux.(Pycharm,2020) Για την δημιουργία του πηγαίου κώδικα του προγράμματος χρησιμοποιήθηκε στο έπακρο η λειτουργικότητα του Pycharm. Η προτίμηση του Pycharm έγινε χάρη στη φιλική διεπαφή ως προς τον χρήστη, την λειτουργικότητά του αλλά και την ευκολία εισαγωγής πρόσθετων βιβλιοθηκών της python. Το λογισμικό Pycharm εκτελέστηκε σε λογισμικό περιβάλλον **Windows 10 64bit.** Η έκδοση του Pycharm που χρησιμοποιήθηκε ηταν η **Pycharm IDE 2020** (PyCharm, 2023)**.**

Εικόνα 9 [ΚΕΦ.1.] Λογότυπο της Python



Εικόνα 10 [ΚΕΦ.1.] Ανάπτυξη προγράμματος με τη χρήση PyCharm2020

1.2.2. Δημιουργία γραφικού περιβάλλοντος της εφαρμογής διαχείρισης αρχείων, με το tkinter

Το GUI(Graphical User Interface) σημαίνει Γραφική διεπαφή χρήστη και αναφέρεται σε προγράμματα υπολογιστών που παρέχουν ένα οπτικό μέσο για τους χρήστες να αλληλοεπιδρούν με μια υποκείμενη εφαρμογή ή σύστημα. Για παράδειγμα, τα GUI σε εφαρμογές μας επιτρέπουν να αλληλοεπιδράσουμε με διαφορετικές λειτουργίες μέσω της οθόνης, τις οποίες μπορούμε να πληκτρολογήσουμε, να πατήσουμε και να σύρουμε επάνω. Σε αυτό το κεφάλαιο, εξετάζουμε ακριβώς τι είναι ο προγραμματισμός Python GUI με τη βοήθεια του tkinter. Το tkinter είναι ένα πλαίσιο γραφικών διεπαφής - διεπαφής πλατφορμών που ενσωματώνεται στην τυπική

βιβλιοθήκη της Python. Δεδομένου ότι είναι ένα πλαίσιο πολλαπλών πλατφορμών, ο ίδιος κώδικας μπορεί να χρησιμοποιηθεί σε οποιοδήποτε λειτουργικό σύστημα, όπως Linux, macOS και Windows. Το Tkinter παρέχει μια αντικειμενοστραφή διεπαφή στην εργαλειοθήκη Tk GUI. Σε σύγκριση με άλλες διαθέσιμες επιλογές, το Tkinter είναι ελαφρύ και εύκολο στη χρήση. Ως εκ τούτου, είναι η καλύτερη επιλογή για γρήγορη κατασκευή εφαρμογών που μπορούν να λειτουργούν σε πολλαπλές πλατφόρμες και δεν απαιτούν μοντέρνα εμφάνιση. Το tkinter παρέχει μια ποικιλία κοινών στοιχείων GUI που μπορούν να χρησιμοποιηθούν για τη δημιουργία διεπαφών. Αυτά τα στοιχεία περιλαμβάνουν κουμπιά, μενού και διάφορα είδη πεδίων καταχώρισης και περιοχές εμφάνισης.

1.2.3. Βάση δεδομένων και δημιουργία με την χρήση του xampp.

Το XAMPP είναι ένας από τους ευρέως χρησιμοποιούμενους διακομιστές ιστού πολλαπλών πλατφορμών, ο οποίος βοηθά τους προγραμματιστές να δημιουργήσουν και να δοκιμάσουν τα προγράμματά τους σε έναν τοπικό διακομιστή ιστού. Αναπτύχθηκε από τους Apache Friends και ο εγγενής πηγαίος κώδικας του μπορεί να αναθεωρηθεί ή να τροποποιηθεί από το κοινό. Αποτελείται από Apache HTTP Server, MariaDB και διερμηνέα για τις διάφορες γλώσσες προγραμματισμού όπως η PHP και η Perl. Είναι διαθέσιμο σε 11 γλώσσες και υποστηρίζεται από διαφορετικές πλατφόρμες όπως το πακέτο IA-32 των Windows & το πακέτο x64 των macOS και Linux. Το XAMPP είναι μια συντομογραφία όπου το **X** σημαίνει **Cross-Platform**, το **A** σημαίνει **Apache**, το **M** σημαίνει **MYSQL** και τα **P** σημαίνουν **PHP** και **Perl** αντίστοιχα. Είναι ένα πακέτο λύσεων ιστού ανοιχτού κώδικα που περιλαμβάνει διανομή Apache για πολλούς διακομιστές και εκτελέσιμα στοιχεία γραμμής εντολών μαζί με ενότητες όπως διακομιστής Apache, MariaDB , PHP και Perl (Xampp, 2023).

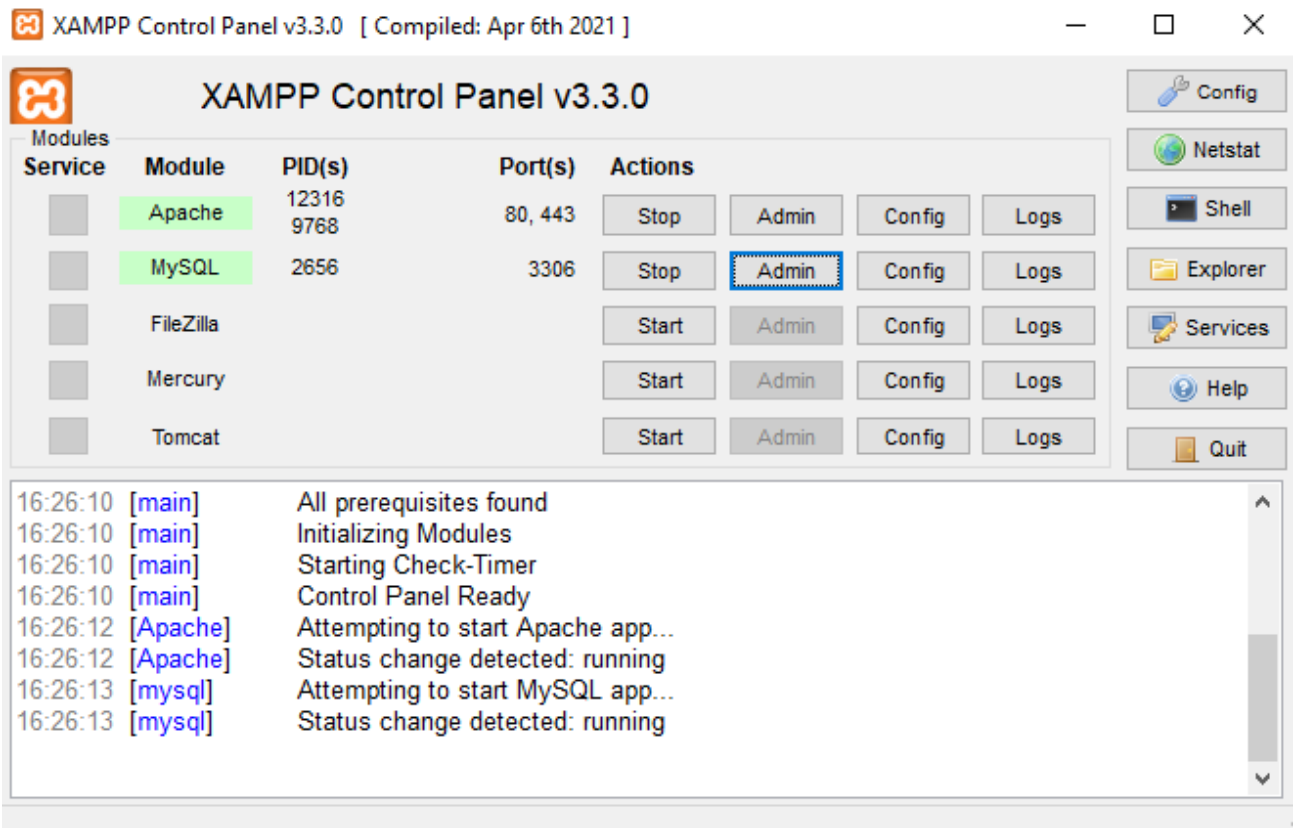
Το XAMPP βοηθά έναν τοπικό κεντρικό υπολογιστή ή διακομιστή να δοκιμάσει τον ιστότοπο και τους πελάτες του μέσω υπολογιστών και φορητών υπολογιστών πριν τον αποδεσμεύσει στον κύριο διακομιστή. Είναι μια πλατφόρμα που παρέχει ένα κατάλληλο περιβάλλον για τη δοκιμή και την επαλήθευση της λειτουργίας των έργων που βασίζονται σε Apache, Perl, MySQL βάση δεδομένων και PHP μέσω του

συστήματος του ίδιου του κεντρικού υπολογιστή. Μεταξύ αυτών των τεχνολογιών, η Perl είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη Ιστού, η PHP είναι μια γλώσσα δέσμης ενεργειών υποστήριξης και η MariaDB είναι η πιο ζωντανή βάση δεδομένων που αναπτύχθηκε από τη MySQL. Στοιχεία που αποτελούν μέρος αυτής της συλλογής λογισμικού επεξηγούνται παρακάτω.

1. Cross-Platform: Διαφορετικά τοπικά συστήματα έχουν διαφορετικές διαμορφώσεις λειτουργικών συστημάτων εγκατεστημένα σε αυτά. Το στοιχείο cross-platform έχει συμπεριληφθεί για να αυξηθεί η χρησιμότητα και το κοινό για αυτό το πακέτο διανομών Apache. Υποστηρίζει διάφορες πλατφόρμες όπως πακέτα Windows, Linus και MAC OS.
2. Apache: Είναι ένας διακομιστής ιστού HTTP, ένας δια-πλατφορμικός διακομιστής. Χρησιμοποιείται παγκοσμίως για την παράδοση περιεχομένου ιστού. Η εφαρμογή διακομιστή έχει γίνει δωρεάν για εγκατάσταση και χρησιμοποιείται για την κοινότητα των προγραμματιστών υπό την αιγίδα του Apache Software Foundation. Ο απομακρυσμένος διακομιστής του Apache παραδίδει τα ζητούμενα αρχεία, εικόνες και άλλα έγγραφα στον χρήστη.
3. MariaDB: Αρχικά, το MySQL DBMS ήταν μέρος του XAMPP, αλλά τώρα έχει αντικατασταθεί από το MariaDB. Είναι ένα από τα πιο ευρέως χρησιμοποιούμενα σχεσιακά DBMS, που αναπτύχθηκε από την MySQL. Προσφέρει διαδικτυακές υπηρεσίες αποθήκευσης, χειρισμού, ανάκτησης, διευθέτησης και διαγραφής δεδομένων.
4. PHP: Είναι η γλώσσα δέσμης ενεργειών υποστήριξης που χρησιμοποιείται κυρίως για την ανάπτυξη Ιστού. Η PHP επιτρέπει στους χρήστες να δημιουργούν δυναμικούς ιστότοπους και εφαρμογές. Μπορεί να εγκατασταθεί σε κάθε πλατφόρμα και υποστηρίζει μια ποικιλία συστημάτων διαχείρισης βάσεων δεδομένων. Υλοποιήθηκε με χρήση γλώσσας C. Η PHP σημαίνει Επεξεργαστής Υπερκειμένου . Λέγεται ότι προέρχεται από τα εργαλεία προσωπικής αρχικής σελίδας, γεγονός που εξηγεί την απλότητα και τη λειτουργικότητά του.
5. Perl: Είναι ένας συνδυασμός δύο δυναμικών γλωσσών υψηλού επιπέδου, δηλαδή της Perl 5 και της Perl 6. Η Perl μπορεί να εφαρμοστεί για την εύρεση

λύσεων για προβλήματα που βασίζονται στη διαχείριση συστήματος, την ανάπτυξη ιστού και τη δικτύωση. Η Perl επιτρέπει στους χρήστες της να προγραμματίζουν δυναμικές εφαρμογές Ιστού. Είναι πολύ ευέλικτο και στιβαρό.

6. phpMyAdmin: Είναι ένα εργαλείο που χρησιμοποιείται για την αντιμετώπιση του MariaDB. Η έκδοση 4.0.4 του χρησιμοποιείται αυτήν τη στιγμή στο XAMPP. Η διαχείριση του DBMS είναι ο κύριος ρόλος του.
7. OpenSSL: Είναι η υλοποίηση ανοιχτού κώδικα του Secure Socket Layer Protocol και Transport Layer Protocol. Επί του παρόντος, η έκδοση 0.9.8 είναι μέρος του XAMPP.
8. Πίνακας ελέγχου XAMPP: Είναι ένας πίνακας που βοηθά στη λειτουργία και τη ρύθμιση σε άλλα εξαρτήματα του XAMPP. Η έκδοση 3.2.1 είναι η πιο πρόσφατη ενημέρωση. Μια λεπτομερής περιγραφή του πίνακα ελέγχου θα γίνει στην επόμενη ενότητα του σεμιναρίου.
9. Webalizer: Είναι μια λύση λογισμικού Web Analytics που χρησιμοποιείται για αρχεία καταγραφής χρηστών και παρέχει λεπτομέρειες σχετικά με τη χρήση.
10. Mercury: Είναι ένα σύστημα μεταφοράς αλληλογραφίας και η τελευταία του έκδοση είναι η 4.62. Είναι ένας διακομιστής αλληλογραφίας, ο οποίος βοηθά στη διαχείριση των μηνυμάτων στον ιστό.
11. Tomcat: Η έκδοση 7.0.42 χρησιμοποιείται αυτήν τη στιγμή στο XAMPP. Είναι ένα servlet που βασίζεται στην JAVA για την παροχή λειτουργιών JAVA.
12. Filezilla: Είναι ένας διακομιστής πρωτοκόλλου μεταφοράς αρχείων, ο οποίος υποστηρίζει και διευκολύνει τις λειτουργίες μεταφοράς που εκτελούνται σε αρχεία. Η πρόσφατα ενημερωμένη έκδοσή του είναι 0.9.41.

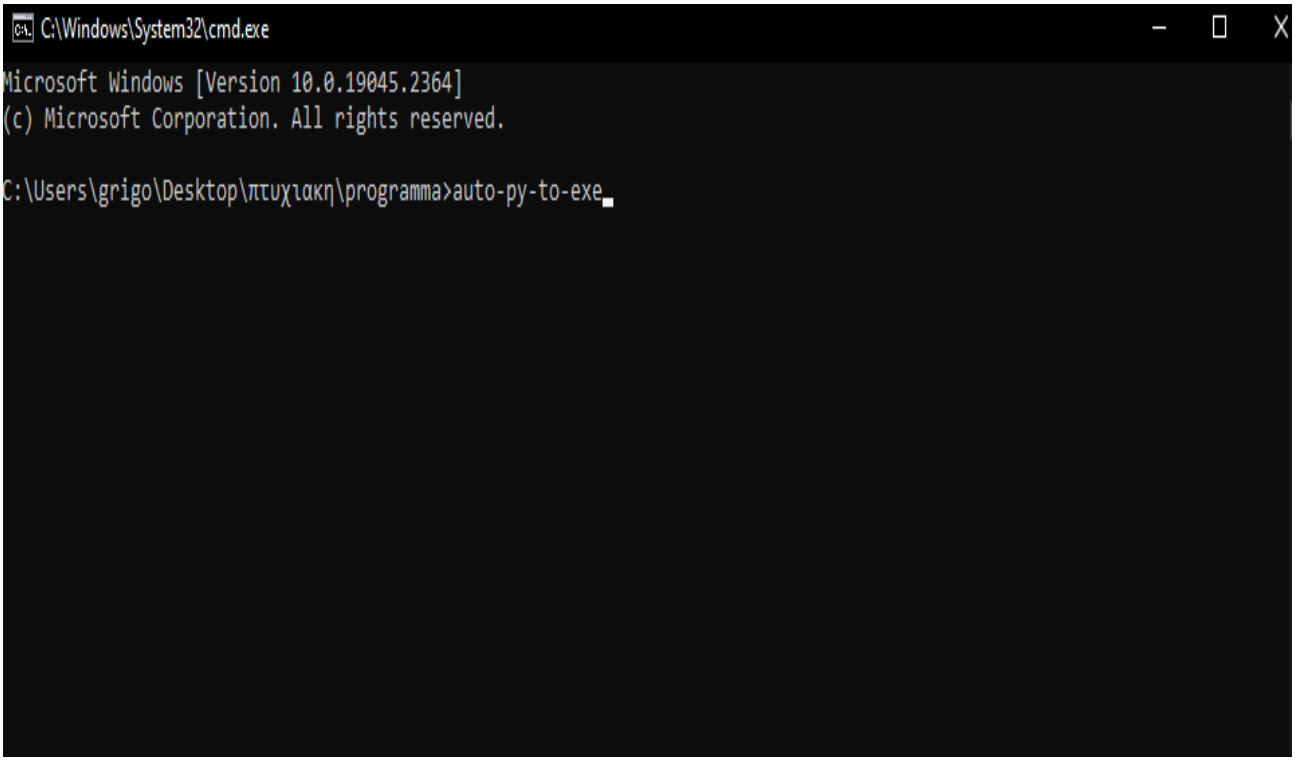


Εικόνα 11 [ΚΕΦ.1.] Πίνακας διαχείρισης του XAMPP

1.2.4. Δημιουργία εκτελέσιμου αρχείου με την χρήση του auto-py-to-exe.

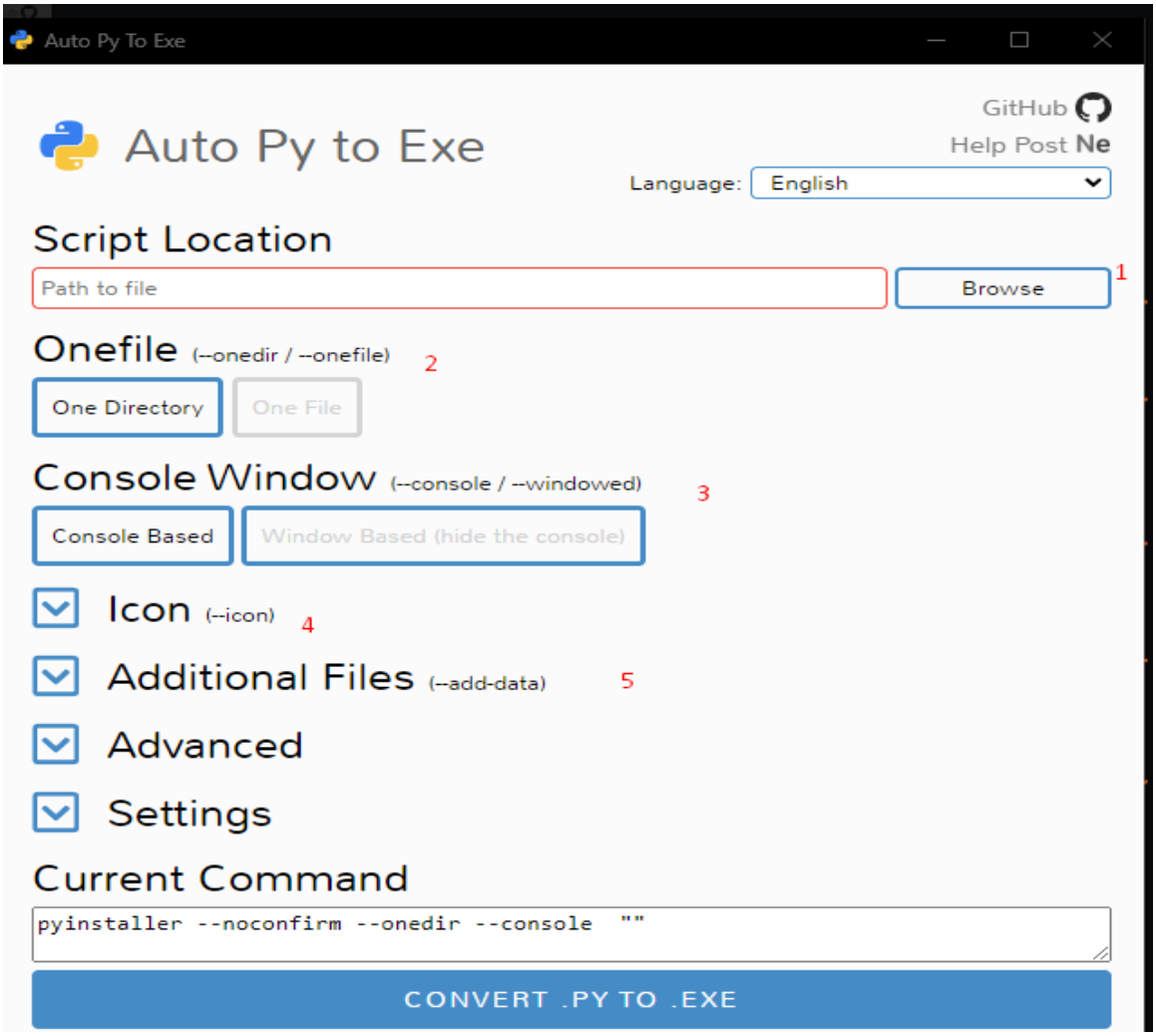
Το auto-py-to-exe είναι μια ειδική βιβλιοθήκη της Python που μπορεί να μετατρέψει ένα αρχείο με κατάληξη .py σε αρχείο .exe δηλαδή σε εκτελέσιμο αρχείο. Το αρχείο που παράγεται από την βιβλιοθήκη αυτή είναι μια μεταγλωττισμένη έκδοση του πηγαίου κώδικα και όχι ο πραγματικός πηγαίος κώδικας. Αυτό βοηθά στην αποτροπή κλοπής του κώδικα. Παρακάτω θα εξηγηθεί η εγκατάστασή του καθώς και η χρήση του. Οι απαιτήσεις του auto-py-to-exe πριν την εγκατάσταση είναι: Να υπάρχει εγκατεστημένη η Python καθώς επίσης και το περιήγησης ιστού google chrome. Ο λόγος για τον οποίο χρειάζεται η εγκατάσταση του google chrome είναι διότι αλληλεπιδράτε με το auto-py-to-exe μέσω ενός παραθύρου του προγράμματος περιήγησης (AutoPyToExe, 2023). Η μέθοδος για την εγκατάσταση της βιβλιοθήκης γίνεται χρησιμοποιώντας στη γραμμή εντολών την παρακάτω εντολή.

Στη συνέχεια εφόσον εγκατασταθεί επιτυχώς απλώς πρέπει να πληκτρολογηθεί στην γραμμή εντολών. Για να λειτουργήσει αυτό προτείνεται να προστεθεί η διαδρομή του αρχείου Python exe στο Path.



Εικόνα 12 [ΚΕΦ.1.] Εγκατάστασης της βιβλιοθήκης auto-py-to-exe

Αφού ολοκληρωθεί θα πρέπει να ανοίξει το παρακάτω περιβάλλον διεπαφής. Όπου θα γίνει η μετατροπή των αρχείων σε εκτελέσιμα. Καθώς επίσης και ο τρόπος λειτουργίας του.



Εικόνα 13 [ΚΕΦ.1.] Περιβάλλον Διεπαφής του Auto Py To Exe

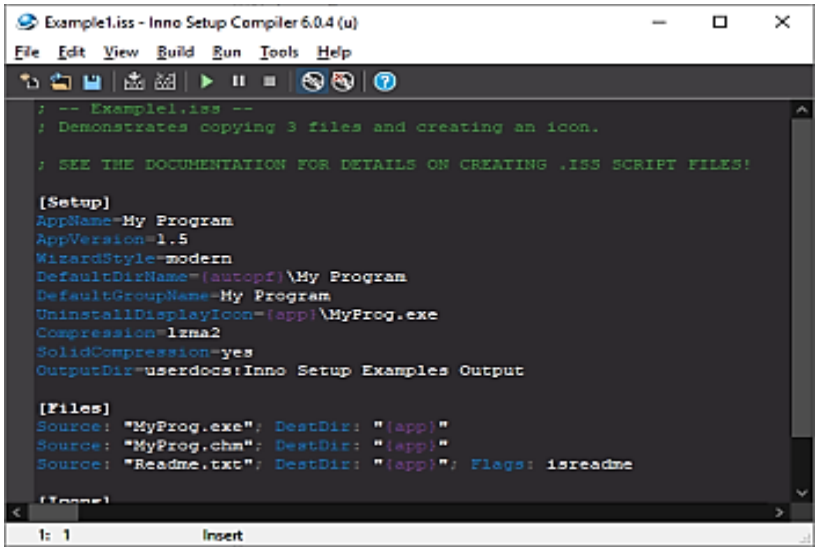
1. Script Location: Επιλέγεται η διαδρομή όπως και το αρχείο python προς μετατροπή. Μπορεί να πληκτρολογηθεί απευθείας η διαδρομή του αρχείου ή χρησιμοποιώντας το κουμπί περιήγησης για την μη αυτόματη επιλογή του.
2. Onefile: Από προεπιλογή , η ρύθμισή του έχει οριστεί σε one-dir . Αυτό συγκεντρώνει όλα τα σχετικά αρχεία και τις εξαρτήσεις σε πολλά αρχεία που βρίσκονται σε έναν κατάλογο. Ενώ η ενεργοποίηση του one-file θα συνδυάσει τα αρχεία σε ένα μεγαλύτερο εκτελέσιμο αρχείο.
3. Console Window: Η επιλογή console based θα ανοίξει ένα παράθυρο κονσόλας όταν εκτελείται το αρχείο. Αυτή χρησιμοποιείται όταν χρειάζεται παράθυρο κονσόλας για να γίνει αλληλεπίδραση με το πρόγραμμα. Εαν, ωστόσο, το πρόγραμμα βασίζεται σε GUI, θα πρέπει να

γίνει μετάβαση σε παραθυρικό περιβάλλον αυτό θα σταματήσει την εμφάνιση παραθύρου κονσόλας. Στην εκδοχή αυτή χρησιμοποιείται η επιλογή Window Based.

4. Icon: Το προεπιλεγμένο εικονίδιο στο εκτελέσιμο της Python δεν είναι όμορφο. Έτσι δίνεται η δυνατότητα εισαγωγής προσαρμοσμένου εικονιδίου χρησιμοποιώντας την διαδρομή του εικονιδίου.
5. Additional Files: Κατά τη δημιουργία λογισμικού, ειδικά σε ενα με GUI ,γίνεται χρήση πρόσθετων όπως εικόνες, αρχεία κειμένου .Στην επιλογή αυτή γίνεται η ομαδοποίησή τους χρησιμοποιώντας τη ρύθμιση προσθήκης φακέλου.

1.2.5. Δημιουργία αρχείου εγκατάστασης για την εφαρμογή με το Inno Setup by Jordan Russel and Martijn Laan.

Στο τελικό στάδιο παραγωγής του προγράμματος εντάσσεται και η δημιουργία του εκτελέσιμου αρχείου εγκατάστασής της εφαρμογής. Αυτό πετυχαίνεται με το εργαλείο Inno Setup. Λογική του βοηθητικού αυτού προγράμματος, είναι η εισαγωγή του εκτελέσιμου αρχείου exe, και η περεταίρω παραμετροποίηση του μέσα από το περιβάλλον εργασίας του. Δίνονται χρήσιμες επιλογές που ο χρήστης μπορεί να χρησιμοποιήσει για το αρχείο εγκατάστασης που θα παράγει, όπως αυτή της ρύθμισης διαφόρων ενεργειών που θα πραγματοποιηθούν στην διάρκεια της εγκατάστασης. Αυτές οι ενέργειες επιτυγχάνονται μέσω scripts. (Inno Setup, 2020)



Εικόνα 14 [ΚΕΦ.1.] Περιβάλλον εργασίας Inno Setup

Κεφάλαιο 2ο: Σχεδιασμός και ανάπτυξη της εφαρμογής CRUD

Για σκοπούς της καλύτερης ανάλυσης και δια δραστικότητας του προγράμματος μας, δόθηκε η ονομασία **CRUD** όπου παραπέμπει στα αρχικά: **Create Read Update Delete**. Στο συγκεκριμένο κεφάλαιο γίνεται ανάλυση της εφαρμογής και παρουσίαση του σχεδιασμού της με την χρήση διαγραμμάτων UML. Το κεφάλαιο ολοκληρώνεται με την ανάπτυξη της βάση δεδομένων που χρησιμοποιήθηκε για την εφαρμογή.

2.1.Ανάλυση της εφαρμογής CRUD

Ανοίγοντας την εφαρμογή εμφανίζεται το αρχικό παράθυρο όπου ο χρήστης μπορεί να εγγραφεί ή να συνδεθεί στο σύστημα. Έπειτα ο κάθε χρήστης θα πρέπει επιλέξει να συνδεθεί συμπληρώνοντας τα στοιχεία του (e-mail, κωδικός) για να μπορέσει να εισέλθει στο περιβάλλον της εφαρμογής. Εφόσον ο χρήστης εισάγει σωστά τα στοιχεία του ανοίγει το παράθυρο υποδοχής όπου μπορεί να διαχειριστεί τα πρόσφατα αρχεία του, να ανεβάσει κάποιο εκ νέου ή να μεταβεί στο κύριο παράθυρο πατώντας όλα τα αρχεία όπου αποτελείται από δύο τμήματα:

- **Όλα τα αρχεία του χρήστη:** Στο τμήματα αυτό εμφανίζεται μια λίστα αρχείων όπου έχει ανεβάσει ο χρήστης και όπου μπορεί να τα αναζητήσει , να τα διαγράψει , να τα κατεβάσει να τροποποιήσει την περιγραφής και την ονομασία τους καθώς επίσης να ανεβάσει ένα νέο αρχείο μεμονωμένα είτε μαζικά.
- **Όλα τα αρχεία άλλων χρηστών:** Εδώ του δίνεται η δυνατότητα να αναζητήσει και επιλέγοντας το επιθυμητό αρχείο να διαβάσει την περιγραφή του καθώς επίσης να το κατεβάσει.

2.3 Διαγράμματα UML της εφαρμογής

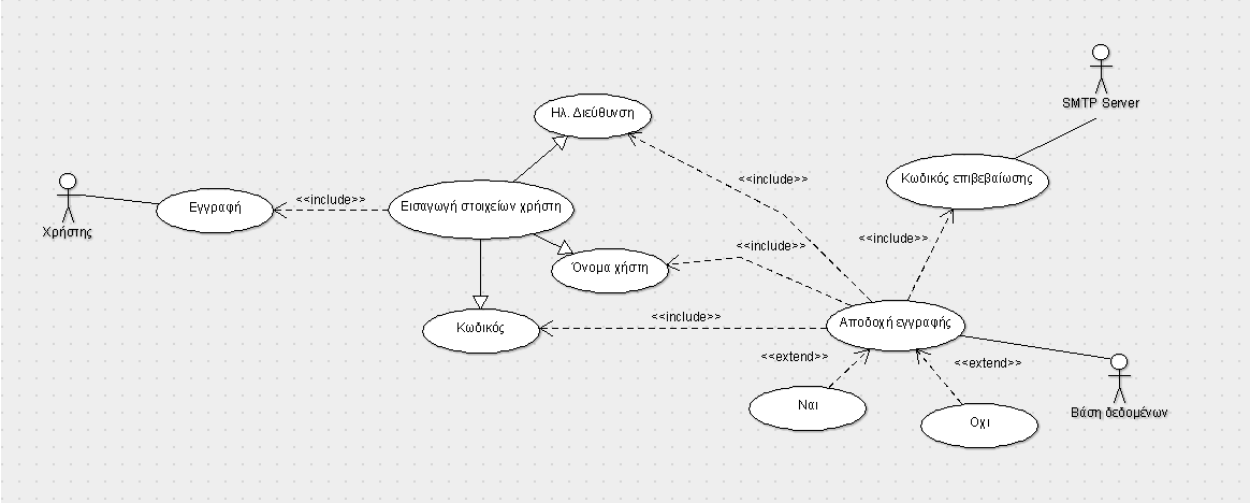
Σε αυτή την ενότητα αναπτύσσονται τα διαγράμματα UML της εφαρμογής CRUD που δημιουργήθηκαν μέσω του προγράμματος ArgoUML. (ArgoUML, 2023)

2.3.1 Διαγράμματα περιπτώσεων χρήσης (use cases diagrams)

Τα διαγράμματα περιπτώσεων χρήσης είναι διαγράμματα λειτουργιών, καθώς απεικονίζουν τις βασικές λειτουργίες του συστήματος – δηλαδή, τι μπορούν να

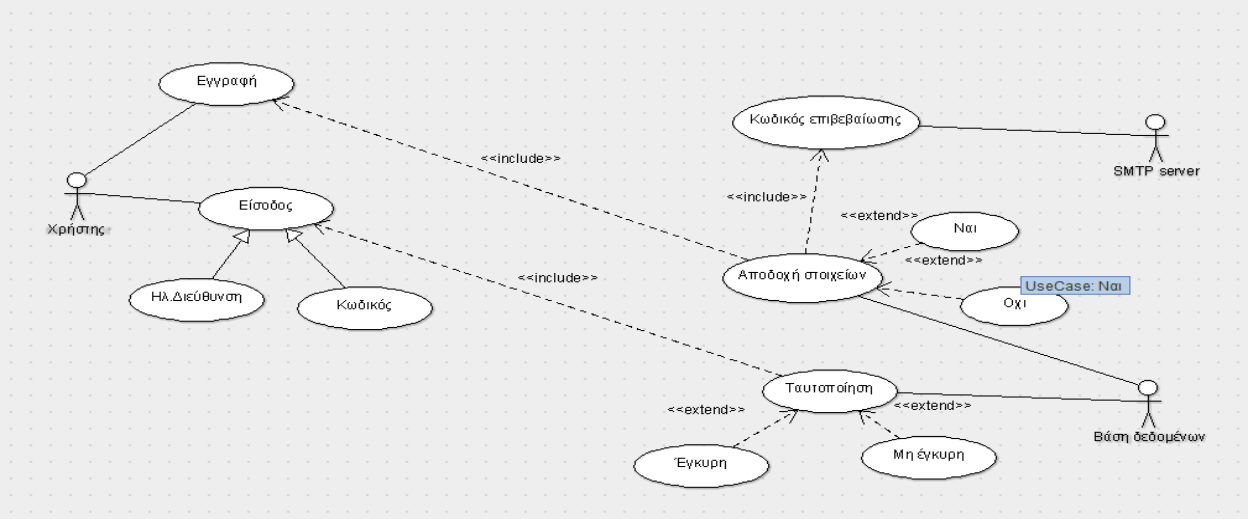
κάνουν οι χρήστες και πως το σύστημα πρέπει να αποκρίνεται στις ενέργειες τους. (Βασίλης, 2006)

Ο χρήστης για να εισέλθει στο σύστημα θα πρέπει να κάνει πρώτα εγγραφή. Συμπληρώνοντας τα στοιχεία του που απαιτούνται για την εγγραφή του, το σύστημα ελέγχει αν είναι έγκυρα τα στοιχεία ή όχι. Αν τα στοιχεία είναι έγκυρα ο SMTP server αποστέλλει ηλεκτρονικό μήνυμα στην διεύθυνση με κωδικό επιβεβαίωσης OTP όπου ο χρήστης πρέπει να τον εισάγει στο πρόγραμμα σε περίπτωση που είναι σωστός καταχωρείται επιτυχώς στην βάση δεδομένων.



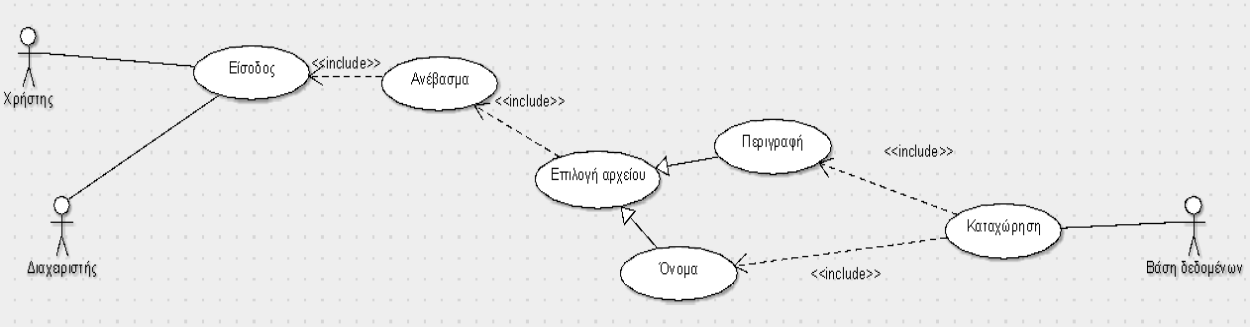
Εικόνα 15[ΚΕΦ.2.]Διάγραμμα περιπτώσεων χρήσης εγγραφή χρήστη

Έπειτα ακολουθεί το διάγραμμα περίπτωσης χρήσης το οποίο είναι για την είσοδο του χρήστη στο σύστημα. Με βασική προϋπόθεση ο χρήστης να έχει εγγραφεί στο σύστημα για να μπορέσει να κάνει την είσοδό του , εισάγει τα στοιχεία του (ηλεκτρονική διεύθυνση και κωδικό). Στην συνέχεια αφού γίνει η ταυτοποίηση στοιχείων εισέρχεται στο σύστημα.



Εικόνα 16[ΚΕΦ.2.] Διάγραμμα περιπτώσεων χρήσης είσοδος χρήστη

Στο επόμενο διάγραμμα βλέπουμε την περίπτωση χρήσης προσθήκης ενός αρχείου είτε από τον χρήστη είτε από τον διαχειριστή του συστήματος. Εφόσον έχει πατηθεί το κουμπί ανέβασμα από το παράθυρο υποδοχής και επιλέξει ένα ή περισσότερα αρχεία ο χρήστης μπορεί να εισάγει περιγραφή και όνομα και να το καταχωρήσει στην βάση δεδομένων.

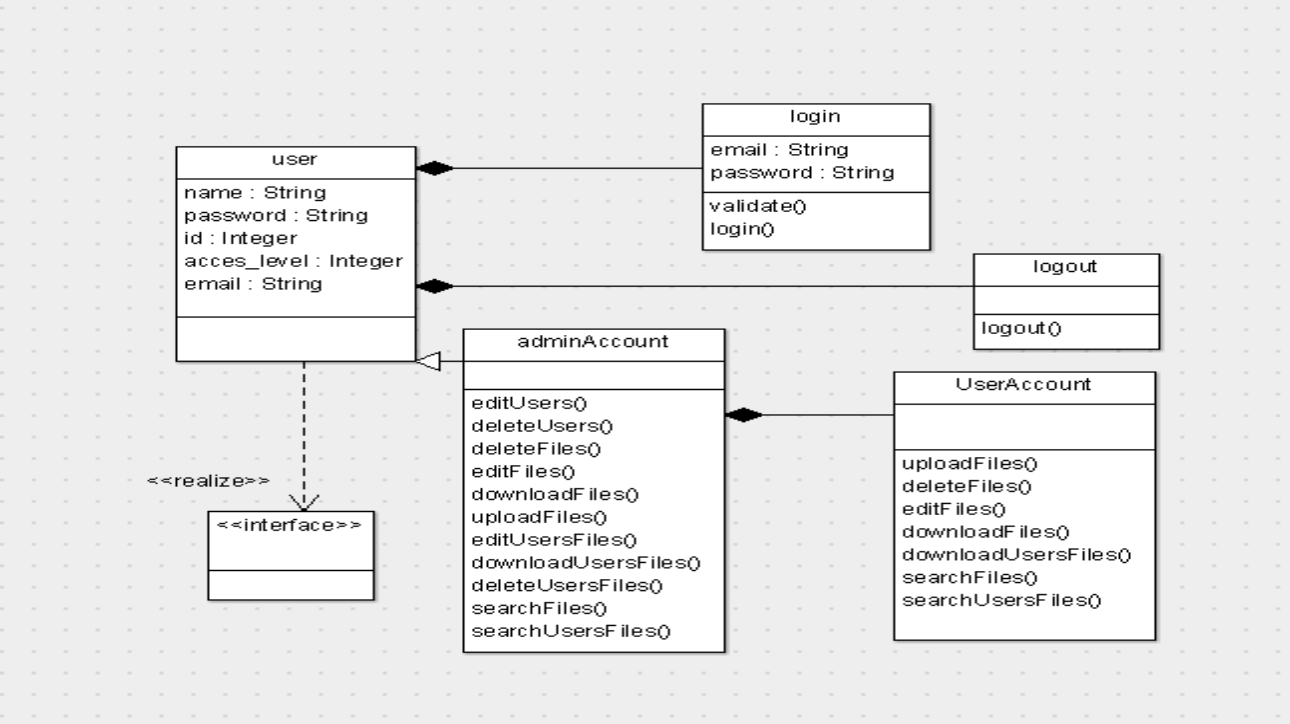


Εικόνα 17 [ΚΕΦ.2.] Διάγραμμα περιπτώσεων χρήσης προσθήκη αρχείου.

2.3.2 Διαγράμματα κλάσεων (class diagrams)

Το διάγραμμα κλάσεων είναι ένα στατικό μοντέλο στο οποίο εμφανίζονται οι κλάσεις καθώς και οι μεταξύ τους σχέσεις που παραμένουν χωρίς καμία αλλαγή στο σύστημα. Παρουσιάζονται κλάσεις, οι οποίες περιλαμβάνουν τόσο συμπεριφορές όσο και καταστάσεις, μαζί με τις σχέσεις μεταξύ των κλάσεων αυτών. (Dennis, 2010) Το διάγραμμα κλάσεων που δημιουργήθηκε αποκλειστικά για την εφαρμογή και αποτελείται από σχέσεις κληρονομικότητας, composition, δηλαδή όταν η μια κλάση

περιέχει μόνιμα την άλλη κλάση και uniComposition, όταν εκτός από το ότι περιέχει η μια κλάση μόνιμα την άλλη, παίρνει και στοιχεία από αυτήν. Επίσης οι κλάσεις αποτελούνται από πεδία και μεθόδους όπως απεικονίζεται παρακάτω. Τέλος υπάρχει το «Interface» που είναι το περιβάλλον της εφαρμογής.



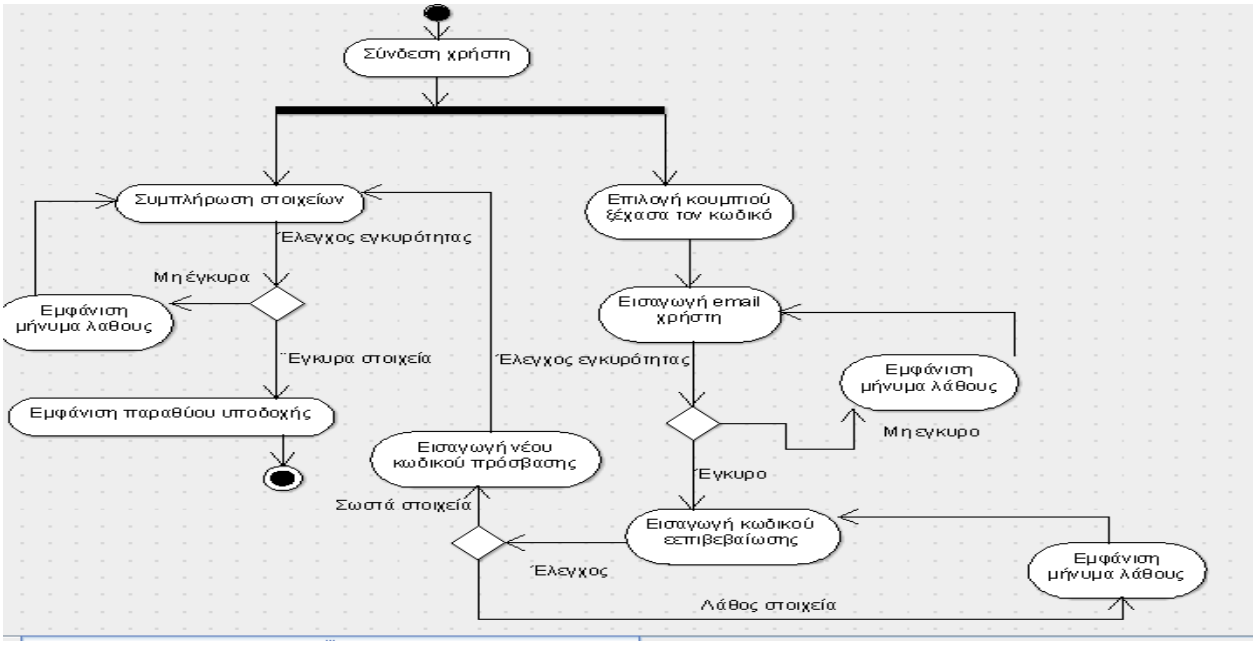
Εικόνα 18 [ΚΕΦ.2.] Διάγραμμα κλάσεων της εφαρμογής CRUD

2.3.3 Διαγράμματα δραστηριοτήτων(activity diagrams).

Τα διαγράμματα δραστηριοτήτων παρέχουν έναν τρόπο για την αποτύπωση των δραστηριοτήτων που λαμβάνουν χώρα στο σύστημα, συμπεριλαμβανομένων και των παράλληλων δραστηριοτήτων που μπορεί να συμβαίνουν σε αυτά. (Βασίλης, 2006)

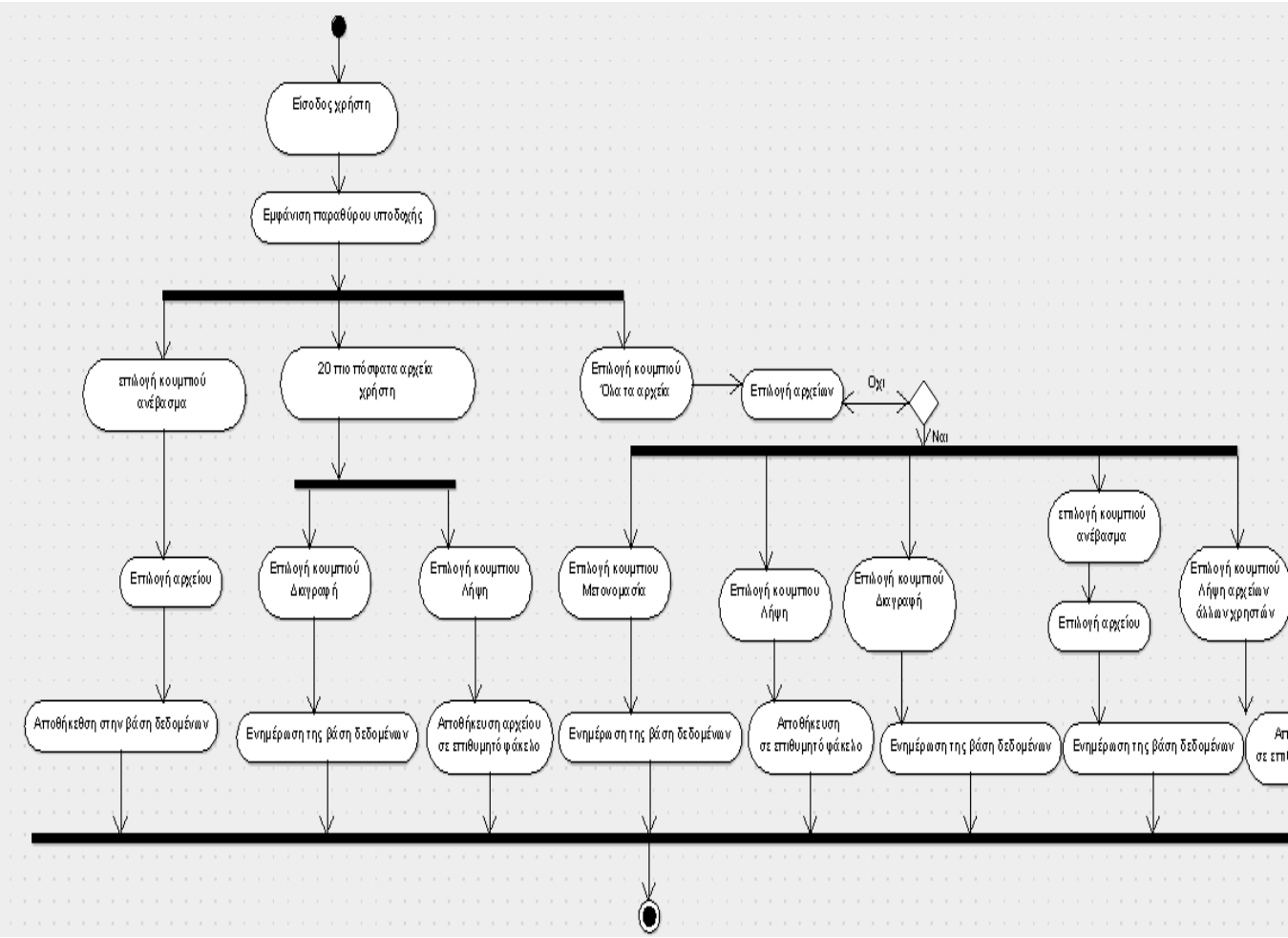
Παρακάτω απεικονίζεται το διάγραμμα δραστηριοτήτων για την είσοδο του χρήστη. Ο χρήστης συμπληρώνει τα στοιχεία του για να γίνει ο έλεγχος εγκυρότητας αυτών έτσι ώστε να εισέλθει στο σύστημα. Σε περίπτωση που είναι λάθος τα στοιχεία μπορεί να τα συμπληρώσει ξανά ή να επιλέξει το κουμπί ξέχασα τον κωδικό όπου ο χρήστης συμπληρώνει το e-mail το οποίο έχει κάνει εγγραφή για να γίνει ο έλεγχος. Εάν είναι σωστό του αποστέλλεται OTP όπου το εάν το εισάγει σωστά μπορεί να αλλάξει τον

κωδικό πρόσβασης, διαφορετικά εάν κάνει λάθος επαναπληκτρολογεί το OTP. Στην περίπτωση που το e-mail είναι λάθος εμφανίζεται σφάλμα και χρήστης μπορεί να εισάγει εκ νέου την ηλεκτρονική του διεύθυνση.



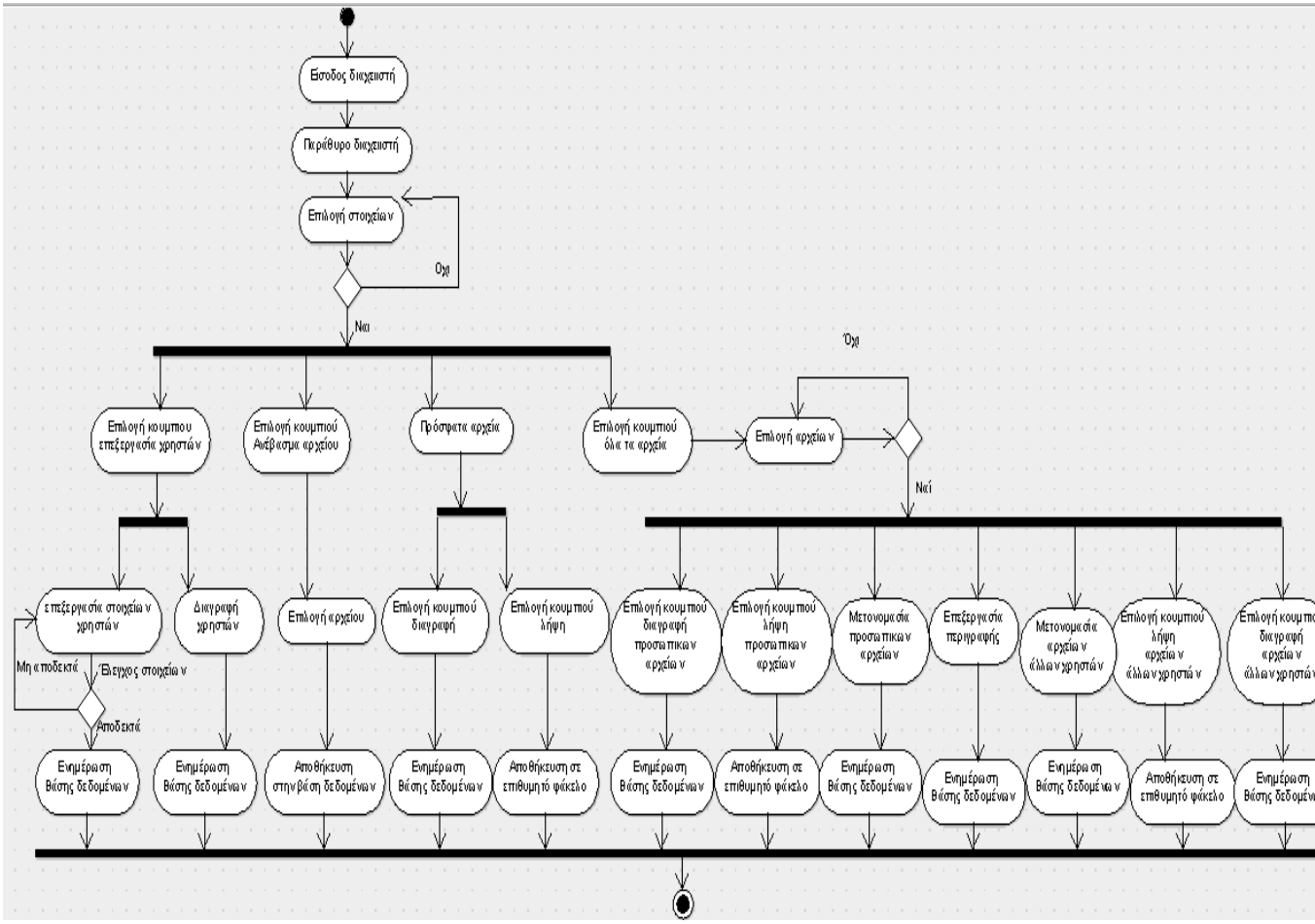
Εικόνα 19 [ΚΕΦ.2.] Διάγραμμα δραστηριοτήτων εισόδου του χρήστη.

Ακολουθεί το διάγραμμα που αφορά τις ενέργειες όπου μπορεί ο χρήστης να πραγματοποιήσει στο σύστημα .Μετά την είσοδό του έχει την δυνατότητα να πραγματοποιήσει τις εξής ενέργειες: Να ανεβάσει, να το κατεβάσει να το τροποποιήσει καθώς επίσης να το διαγράψει ένα αρχείο του ή να κατεβάσει κάποιο αρχείο άλλου χρήστη. Όπως εμφανίζεται στην εικόνα σε κάθε ενέργεια που πραγματοποιείται η βάση δεδομένων ενημερώνεται αντίστοιχα.



Εικόνα 20[ΚΕΦ.2.] Διάγραμμα δραστηριοτήτων δυνατότητες χρήστη

Τα διαγράμματα δραστηριοτήτων ολοκληρώνονται με τις ενέργειες οπου μπορεί να πραγματοποιήσει ο διαχειριστής στο σύστημα. Όπως και ο χρήστης ετσι και ο διαχειριστής μπορεί να πραγματοποιήσει τις ίδιες ενέργειες κάνοντας την είσοδό του αλλά του δίνεται μια ευχέρεια παραπάνω λειτουργειών όπου απεικονίζεται στην παρακάτω εικόνα. Ο διαχειριστής μπορεί να τροποποιήσει ή και να διαγράψει άλλους χρήστες καθώς επίσης και τα αρχεία τους σε κάθε ενέργεια που γίνεται η βάση δεδομένων ενημερώνεται .

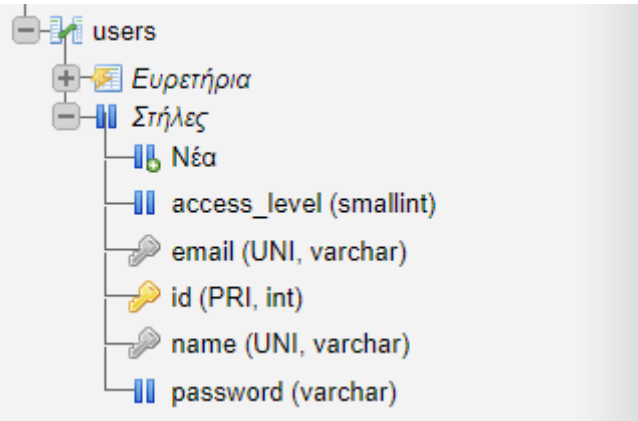


Εικόνα 21 [ΚΕΦ.2.] Διάγραμμα δραστηριοτήτων δυνατότητες διαχειριστή.

2.4 Η βάση δεδομένων της εφαρμογής CRUD

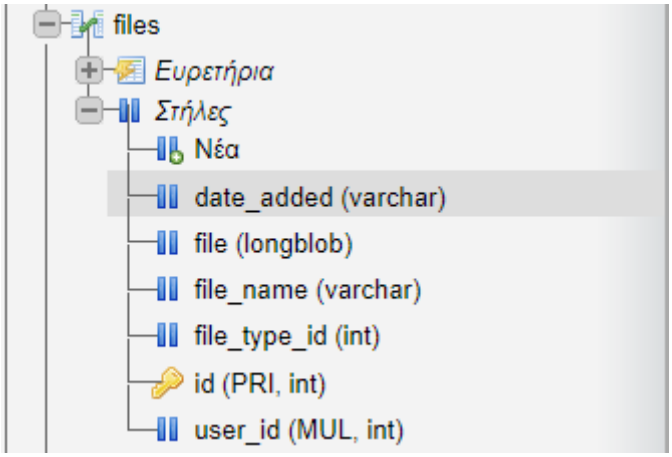
Η βάση δεδομένων δημιουργήθηκε με την βοήθεια του phpmyadmin. Η σύνδεση της βάσης γίνεται μέσω του xampp το οποίο τρέχει τις υπηρεσίες της mysql τοπικά και το πρόγραμμα συνδέεται με την βοήθεια της βιβλιοθήκης mysql.connector. Η βάση δεδομένων αποτελείται από 4 πίνακες:

- **Users:** για την καταχώρηση χρηστών στο σύστημα. Αποτελείται από 5 στήλες στις οποίες εκχωρούνται τα εξής στοιχεία: το επίπεδο πρόσβασης τους, η ηλεκτρονική διεύθυνσή τους, το πρωτεύων κλειδί του πίνακα αποτελεί τον αποκλειστικό κωδικό για κάθε χρήστη (DATE, 1998), το όνομα του εκάστοτε χρήστη καθώς και ο κωδικός.



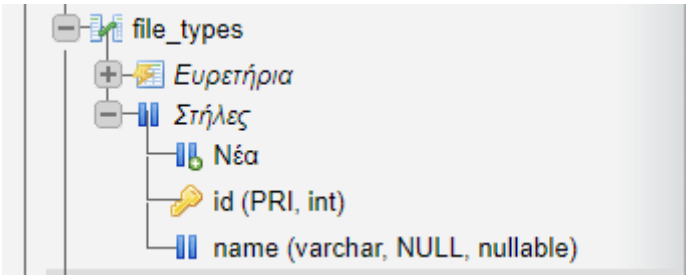
Εικόνα 22 [ΚΕΦ.2.] Πίνακας Users.

- **Files**: όπου χρησιμοποιείται για την αποθήκευση των αρχείων. Αποτελείται από έξι στήλες οι οποίες αποθηκεύουν τις λεπτομέρειες των αρχείων, όπως: την ημερομηνία εκχώρησης, τα αρχεία, το όνομά τους , τον τύπου τους, το πρωτεύων κλειδί και το μεταναστευόμενο πρωτεύων κλειδί από τον πίνακα users οι οποίοι τα ανέβασαν.



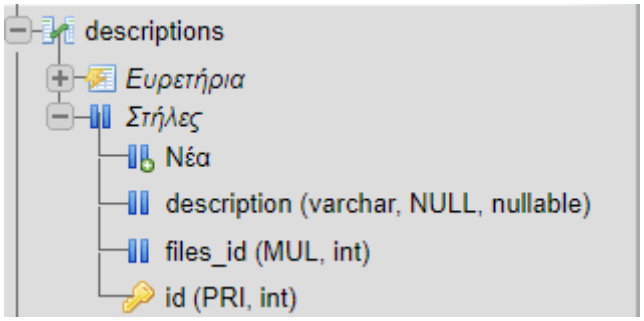
Εικόνα 23 [ΚΕΦ.2.] Πίνακας Files.

- **File_types:** για την καταχώρηση του τύπου των αρχείων. Αποτελείται από τις δύο στήλες το πρωτεύων κλειδί και τον τύπο τους.



Εικόνα 24 [ΚΕΦ.2.] Πίνακας File_types.

- **Descriptions**: για την περιγραφή των αρχείων, όπου περιέχει τρεις στήλες, την περιγραφή τους, το μεταναστευόμενο πρωτεύων κλειδί από τον πίνακα files και το πρωτεύων κλειδί του.



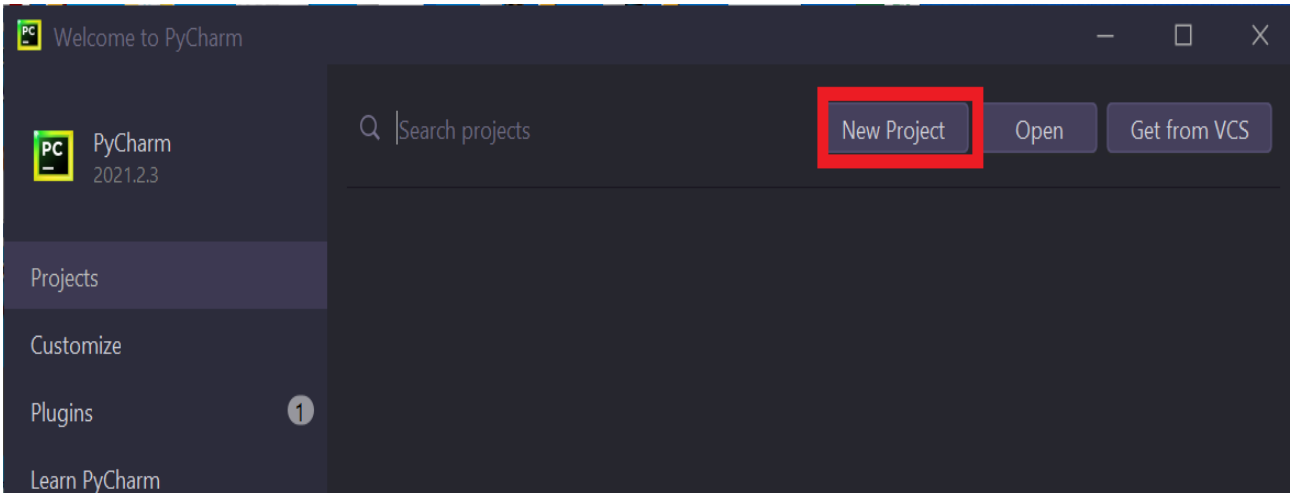
Εικόνα 25 [ΚΕΦ.2.] Πίνακας Descriptions.

Κεφαλαιο 3ο : Η δημιουργία και η λειτουργία της εφαρμογής

Στο τελευταίο κεφάλαιο θα αναπτυχθεί βήμα βήμα η υλοποίηση της εφαρμογής, στη συνέχεια θα γίνει παρουσίαση του τρόπου εγκατάστασης της εφαρμογής σε έναν ηλεκτρονικό υπολογιστή και τέλος θα γίνει αναλυτική επεξήγηση του τρόπου λειτουργίας και των παραμέτρων της.

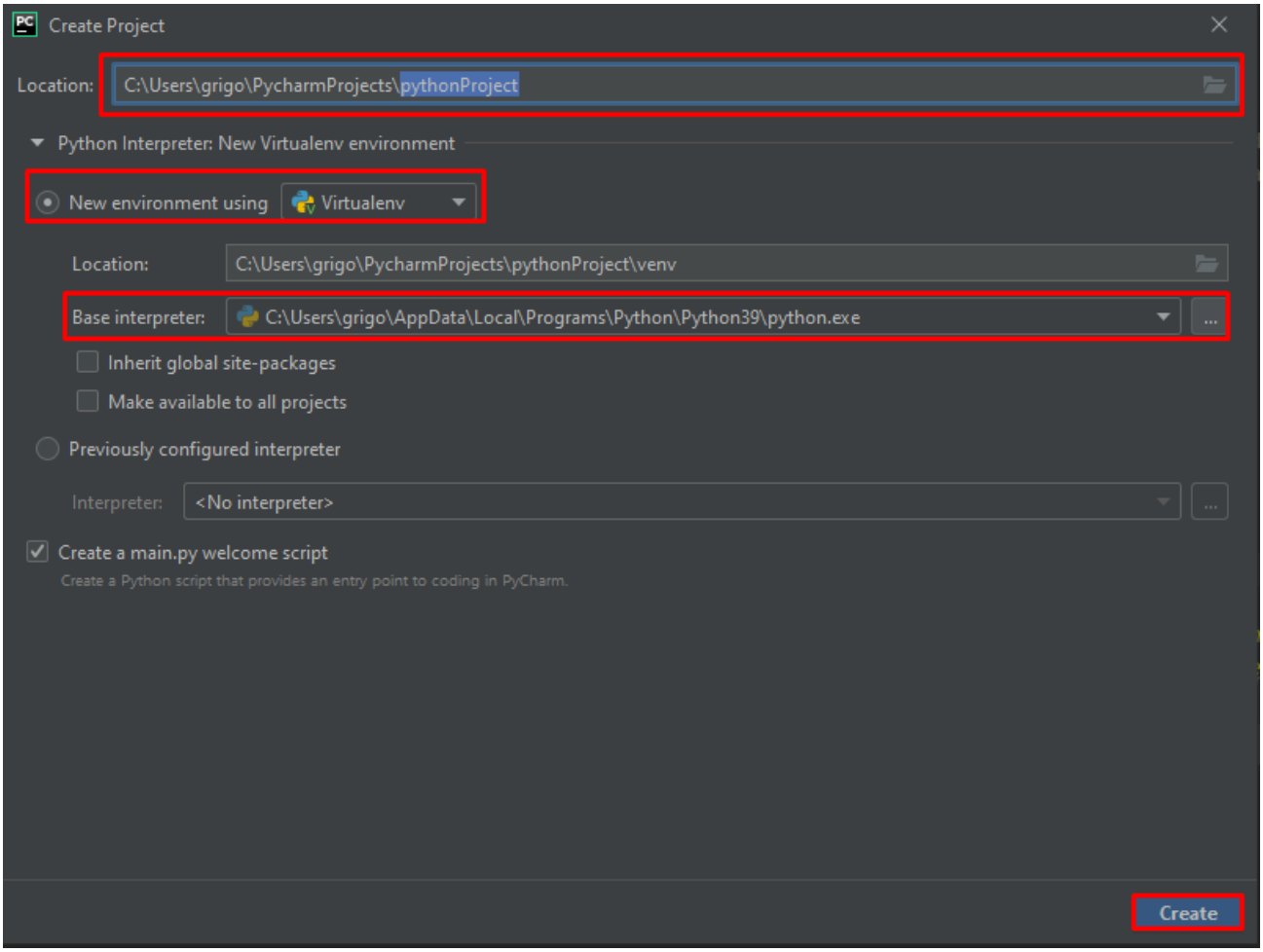
3.1. Η δημιουργία της εφαρμογής CRUD βήμα προς βήμα

Για την δημιουργία της εφαρμογής CRUD χρησιμοποιήθηκε το πρόγραμμα PyCharm, όπου είναι φτιαγμένο σε γλώσσα Python.(PyCharm,2021.2.3). Αρχικά για να δημιουργηθεί η εφαρμογή ένα νέο project επιλέγοντας “New project”.



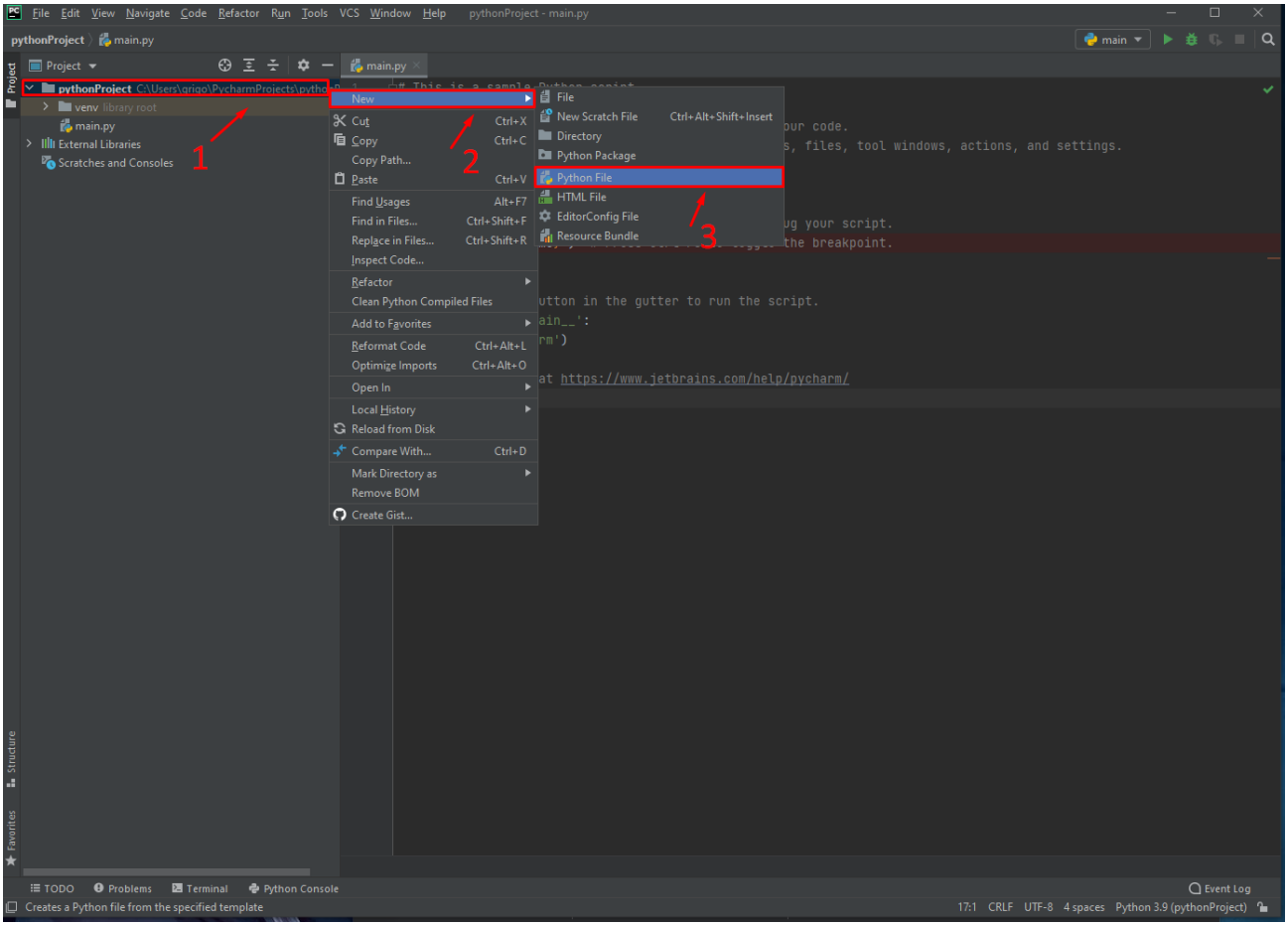
Εικόνα 26 [ΚΕΦ.3.] Δημιουργία νέου project με την χρήση Pycharm2021.2.3.

Στη συνέχεια γίνεται μετάβαση σε νέο παράθυρο όπου γίνεται επιλογή της τοποθεσίας όπου θα δημιουργηθεί το project, καθώς επίσης επιλέγεται το εικονικό περιβάλλον (Virtualenv) και βασικός διερμηνέας. Στο τέλος για να δημιουργηθεί το project πατήθηκε το κουμπί “Create”.



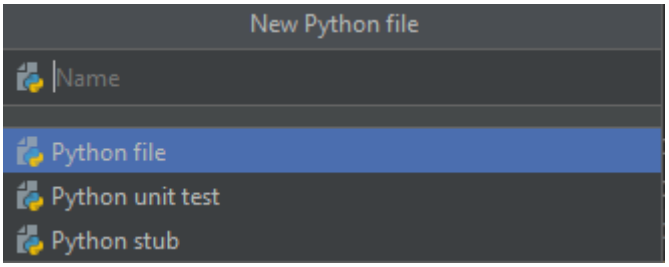
Εικόνα 27 [ΚΕΦ.3.] Δημιουργία εικονικού περιβάλλοντος και επιλογή βασικού διερμηνέα.

Έπειτα δημιουργήθηκαν τρία διαφορετικά αρχεία κάνοντας κάθε φορά τα ακόλουθα βήματα: (1) Δεξί κλικ στο Project > (2) New > (3) Python File.



Εικόνα 28 [ΚΕΦ.3.] Βήματα δημιουργίας αρχείου Python.

Ανοίγοντας το παράθυρο δόθηκαν τα ονόματα των αρχείων (CRUD_main,OTPVerification,Verify) και στη συνέχεια πατήθηκε “Enter”.



Εικόνα 29 [ΚΕΦ.3.] Δημιουργία αρχείου Python.

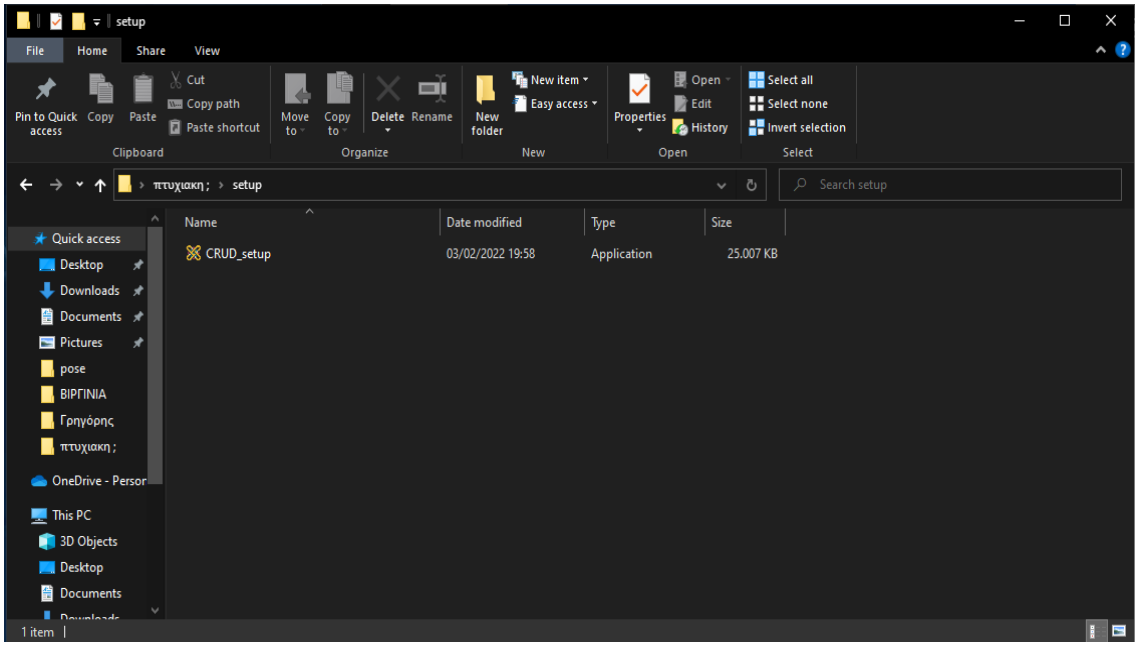
Οι φωτογραφίες οι οποίες χρειάστηκαν στην υλοποίηση του προγράμματος έχουν τοποθετηθεί στον κεντρικό φάκελο του project. Στο αρχείο CRUD_main.py είναι το βασικό αρχείο εκτέλεσης του προγράμματος το οποίο περιέχει: Το γραφικό περιβάλλον της εφαρμογής, τη σύνδεση με τη βάση δεδομένων και όλες τις βασικές λειτουργίες του προγράμματος οι οποίες είναι υπεύθυνες για την ορθή λειτουργία του. Το αρχείο OTPVerification είναι υπεύθυνο για την εισαγωγή κωδικών OTP στα

αναδυόμενα παράθυρα της εγγραφής και της αλλαγής κωδικού πρόσβασης, ενώ παράλληλα ελέγχει τον χρόνο εγκυρότητας του κωδικού(δυο λεπτά) που έχει σταλεί και λανθασμένες προσπάθειες. Σε περίπτωση που είτε γίνουν τρεις λανθασμένες προσπάθειες είτε τελειώσει ο χρόνος των δυο λεπτών, για να ολοκληρωθεί διαδικασία θα πρέπει να αρχίσει εκ νέου. Επιπλέον μέσα από αυτό καλείται το αρχείο Verify.py το οποίο πραγματοποιεί τον έλεγχο επαλήθευσης του εισαγόμενου κωδικού με τον κωδικό όπου στάλθηκε στην ηλεκτρονική διεύθυνση του χρήστη.

3.2. Ο τρόπος εγκατάστασης της εφαρμογής CRUD στους ηλεκτρονικούς υπολογιστές – απαιτήσεις συστήματος.

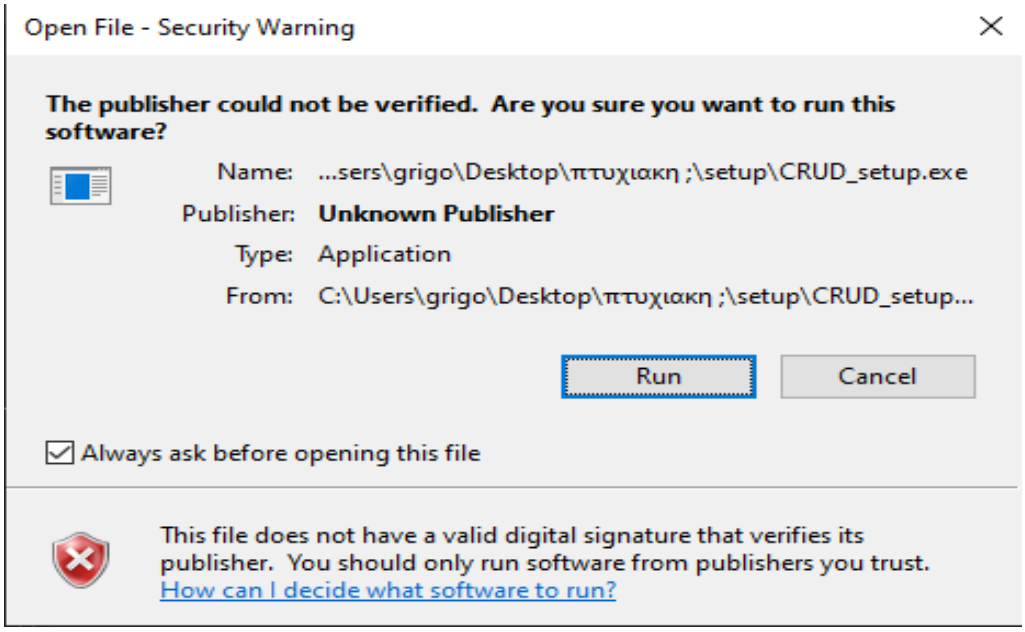
Για την εγκατάσταση της εφαρμογής θα πρέπει να είναι εγκατεστημένος ο server xampp με απαραίτητη προϋπόθεση να υπάρχει ανεβασμένη η βάση δεδομένων σε αυτόν για να είναι δυνατή η σύνδεσή της με την εφαρμογή. Εφόσον γίνουν αύτα εκτελούμε την εφαρμογή για CRUD_setup.exe για να ξεκινήσει η εγκατάσταση. Όπου επιτυγχάνεται με τα εξής βήματα:

1. Επιλογή του εκτελέσιμου αρχείου CRUD_setup.exe.



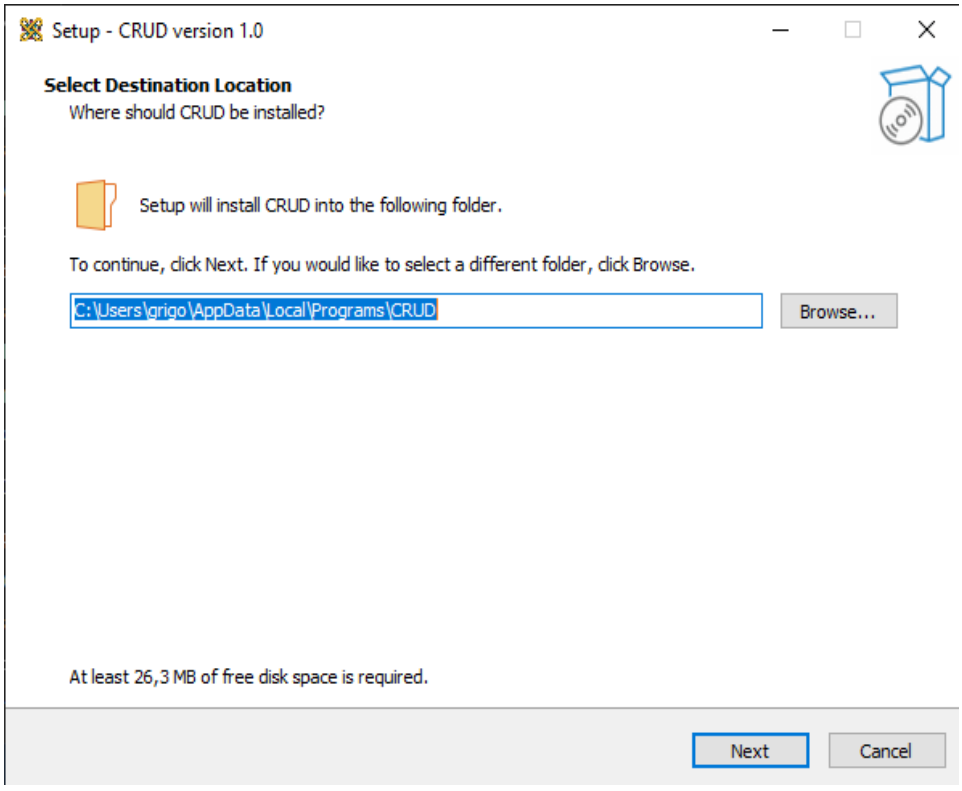
Εικόνα 30 [ΚΕΦ.3.] Επιλογή αρχείου CRUD_setup.

2. Επιλογή του κουμπιού run στο αναδυόμενο παράθυρο.



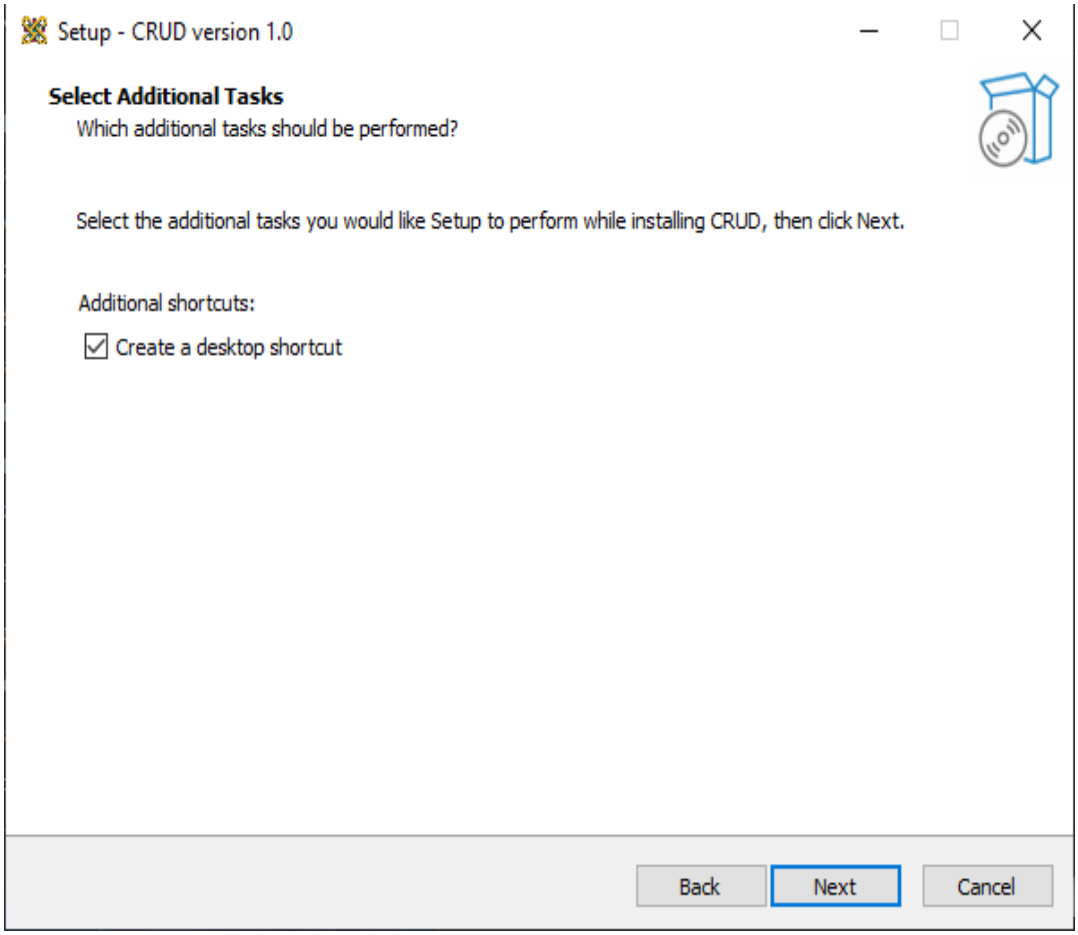
Εικόνα 31 [ΚΕΦ.3.]Εκτέλεση CRUD_setup.

3. Ορισμός σημείου εγκατάστασης της εφαρμογής στον ηλεκτρονικό υπολογιστή και στην συνέχεια επιλογή του κουμπιού “next”.



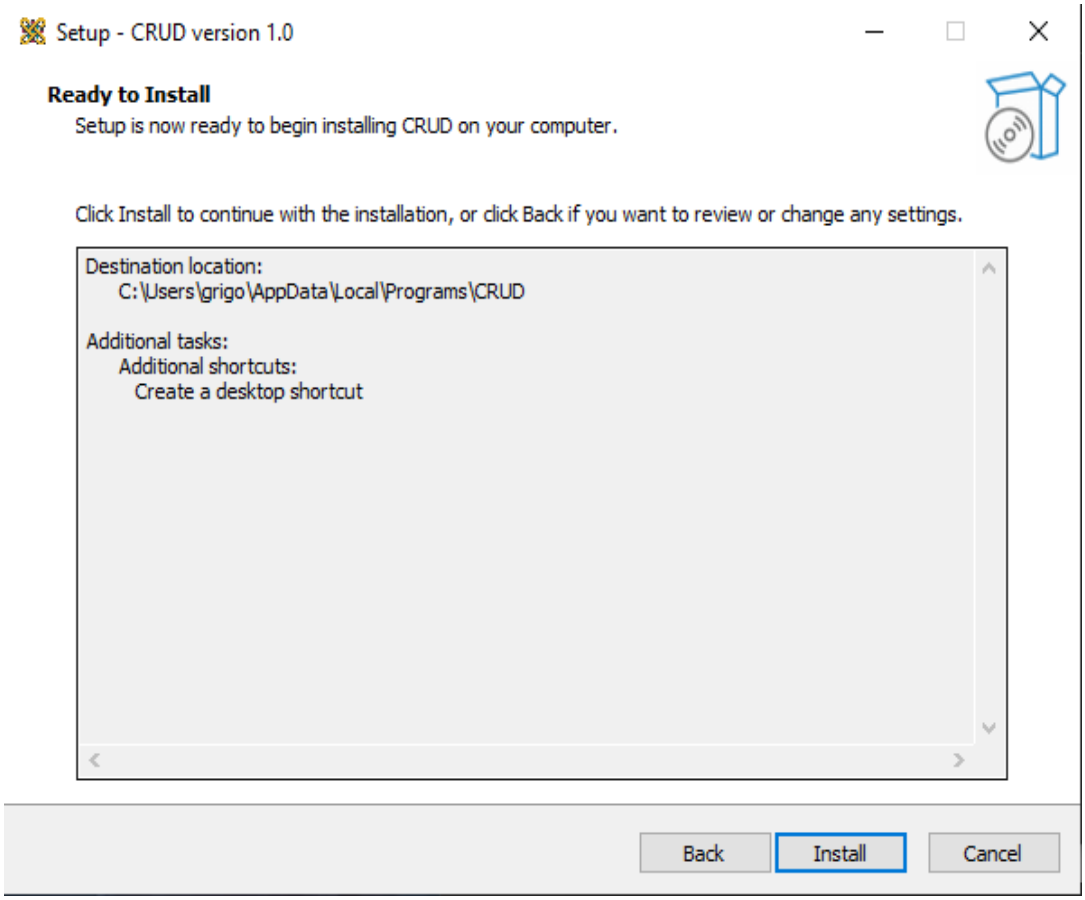
Εικόνα 32[ΚΕΦ.3.]Επιλογή φακέλου εγκατάστασης της εφαρμογής.

4. Στο παράθυρο αυτό έχουμε την δυνατότητα δημιουργίας συντόμευσης στην επιφάνεια εργασίας.



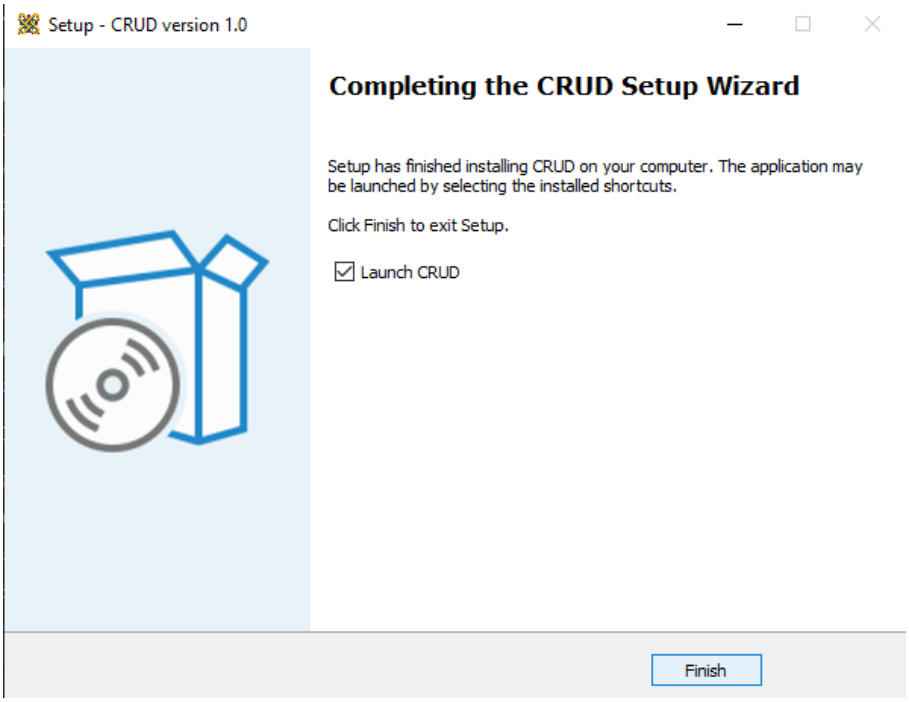
Εικόνα 33[ΚΕΦ.3.] Δημιουργία συντόμευσης της εφαρμογής CRUD.

5. Έπειτα γίνεται ανασκόπηση όλων των ενεργειών μας εφόσον δεν επιθυμούμε κάποια αλλαγή πατάμε το κουμπί ‘’install’’.



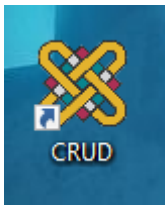
Εικόνα 34 [ΚΕΦ.3.] Εγκατάσταση της εφαρμογής CRUD.

6. Μόλις ολοκληρωθεί η εγκατάσταση επιλέγοντας ‘’Launch CRUD’’ γίνεται η έναρξη της εφαρμογής με αυστηρή προϋπόθεση το xampp να είναι ανοιχτό τέλος η ολοκλήρωση της εγκατάστασης γίνεται πατώντας ‘’finish’’.



Εικόνα 35[ΚΕΦ.3.] Παράθυρο επιτυχής εγκατάστασης.

7. Παρατηρείται παρακάτω ότι μετά την ολοκλήρωση της εγκατάστασης έχει δημιουργηθεί στην επιφάνεια εργασίας η συντόμευση της εφαρμογής.



Εικόνα 36[ΚΕΦ.3.] Εικονίδιο της εφαρμογής CRUD.

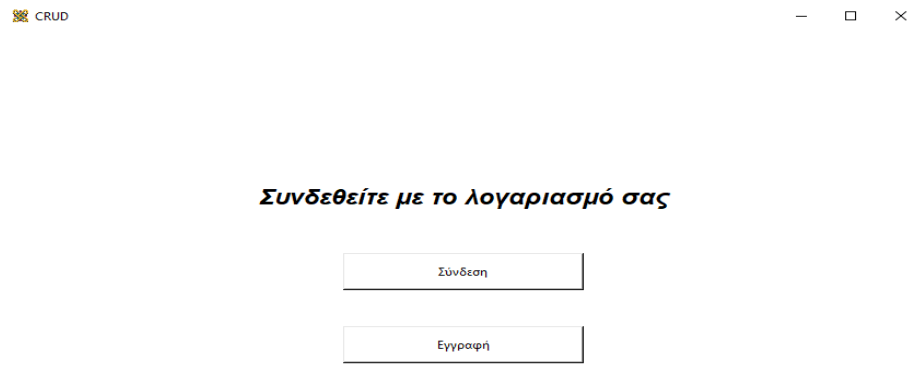
3.2.1. Απαιτήσεις Συστήματος της εφαρμογής AMCS

Οι απαιτήσεις συστήματος της εφαρμογής είναι οι εξής:

- **CPU** - τουλάχιστον 2 πυρήνες.
- **RAM** – τουλάχιστον 4GB μνήμης.
- **Χωρητικότητα Δίσκου** – Αρχείο εγκατεστημένης εφαρμογής 35MB.
- **Λειτουργικό Σύστημα** – Λειτουργικό σύστημα αρχιτεκτονικής X64 και Windows 8,1 ή μετέπειτα.
- **Προ-απαιτούμενα Αρχεία** – Απαιτείται η προεγκατάσταση του προγράμματος xampp καθώς και η έκδοση της python 3.9.

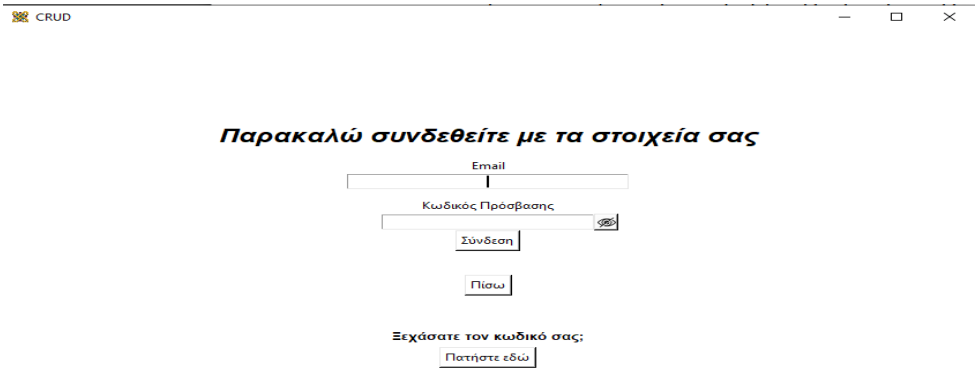
3.3. Αναλυτική παρουσίαση του τρόπου λειτουργίας και των παραμέτρων λειτουργίας της εφαρμογής CRUD

Παρακάτω αναλύεται ο τρόπος λειτουργίας της εφαρμογής. Αρχικά πατώντας την συντόμευση ανοίγει η εφαρμογή και εμφανίζεται το αρχικό παράθυρο της εφαρμογής όπου ο χρήστης μπορεί να συνδεθεί ή να εγγραφεί στο σύστημα .



Εικόνα 37[ΚΕΦ.3.] Αρχικό παράθυρο της εφαρμογής.

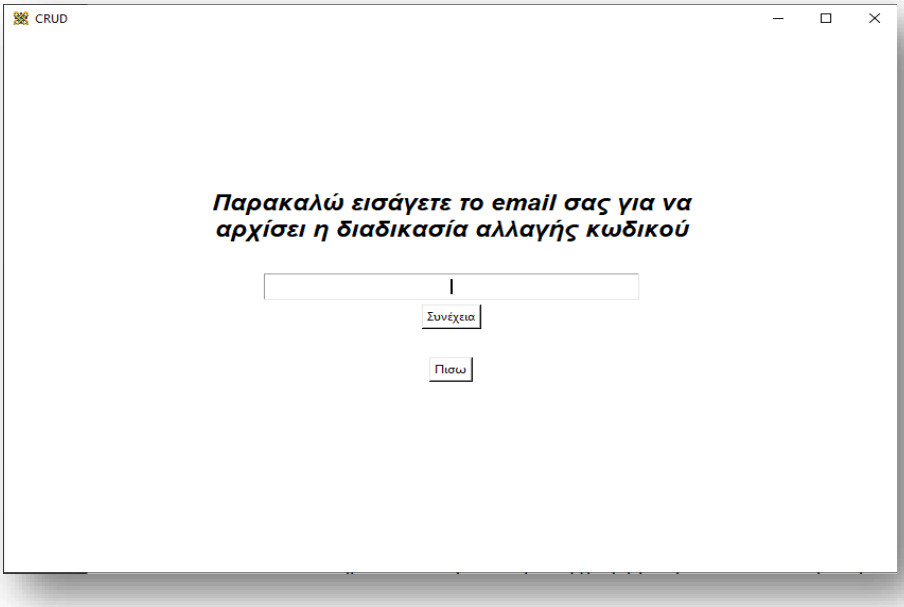
Όταν πατηθεί το κουμπί σύνδεση μεταβαίνουμε στο παράθυρο όπου ο χρήστης μπορεί να εισαχθεί στο σύστημα πληκτρολογώντας την ηλεκτρονική διεύθυνση (email), τον κωδικό (password) .Το σύστημα αναγνωρίζει αυτόματα εαν ο χρήστης που συνδέεται στο σύστημα είναι απλός χρήστης ή διαχειριστής και έπειτα μπορεί να μπει στην εφαρμογή εφόσον τα στοιχεία που έδωσε είναι σωστά.



Εικόνα 38 [ΚΕΦ.3.] Παράθυρο σύνδεσης.

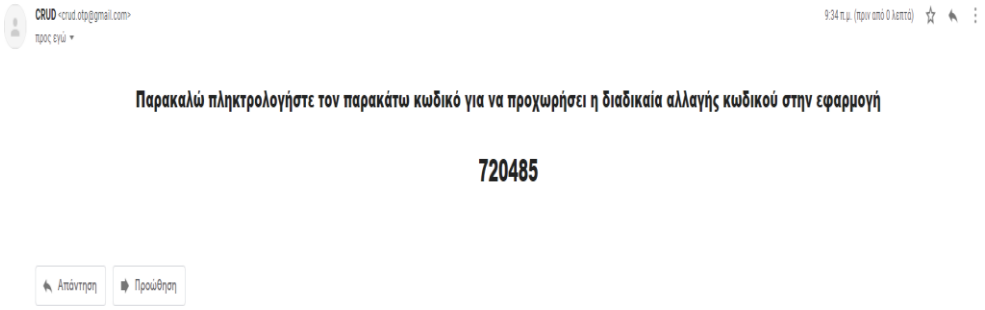
Επίσης σε περίπτωση που έχει ξεχάσει τον κωδικό δίνεται η δυνατότητα ανάκτησής του με τα εξής βήματα:

- Πατώντας ξέχασα τον κωδικό γίνεται μετάβαση σε νέο παράθυρο οπού πρέπει να γίνει η εισαγωγή της ηλεκτρονικής διεύθυνσης του εγγεγραμμένου χρήστη.

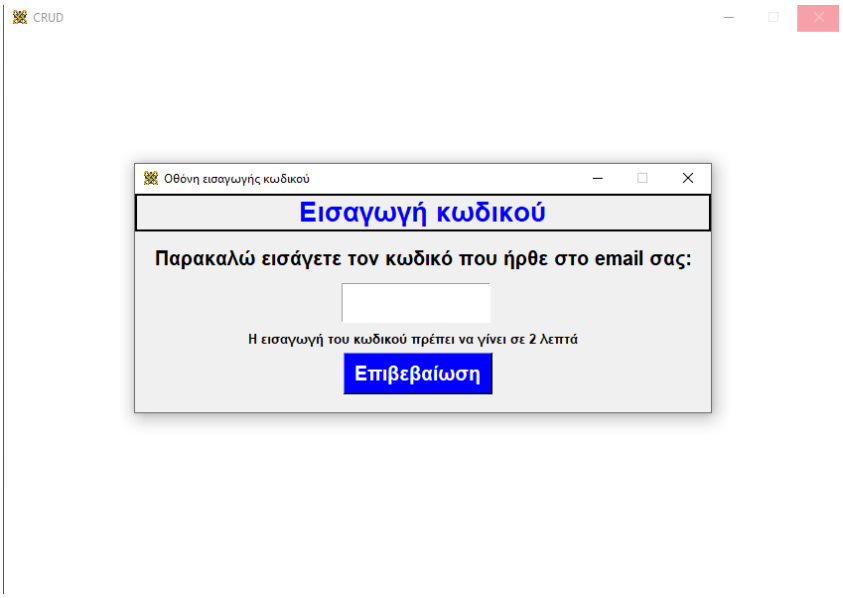


Εικόνα 39[ΚΕΦ.3.] Παράθυρο αλλαγής κωδικού πρόσβασης.

- Αποστέλλεται ο 6ψήφιος κωδικός OTP (One-Time Password) για να συνεχίσει την διαδικασία ανάκτησης.

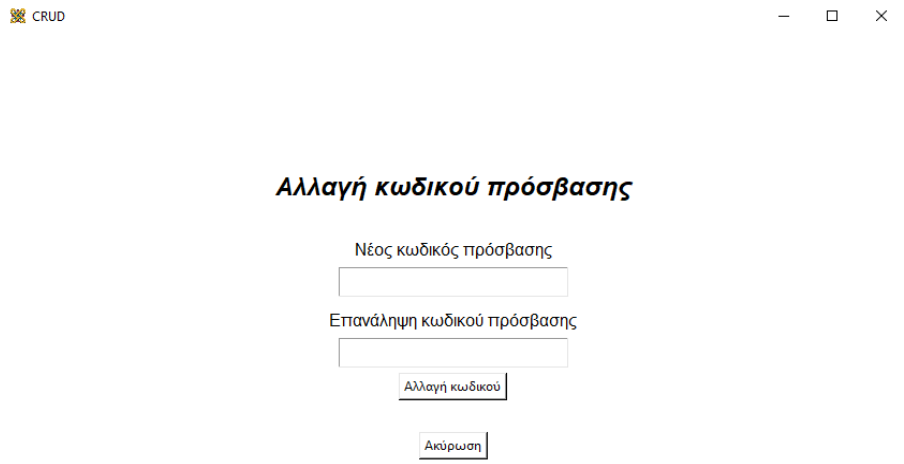


Εικόνα 40[ΚΕΦ.3.] E-mail με OTP.



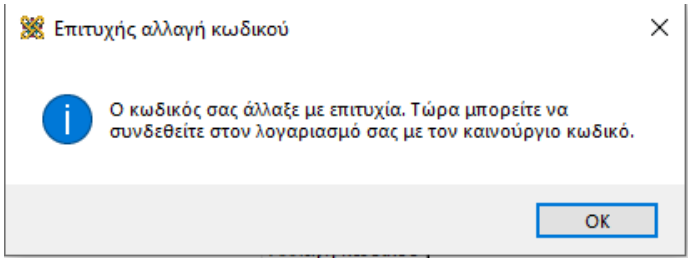
Εικόνα 41[ΚΕΦ.3.] Εισαγωγή OTP.

- Εφόσον το κάνει επιτυχώς ανοίγει το παράθυρο αλλαγής κωδικού πρόσβασης οπού μπορεί να εισάγει νέο κωδικό.



Εικόνα 42 [ΚΕΦ.3.] Παράθυρο αλλαγής κωδικού πρόσβασης.

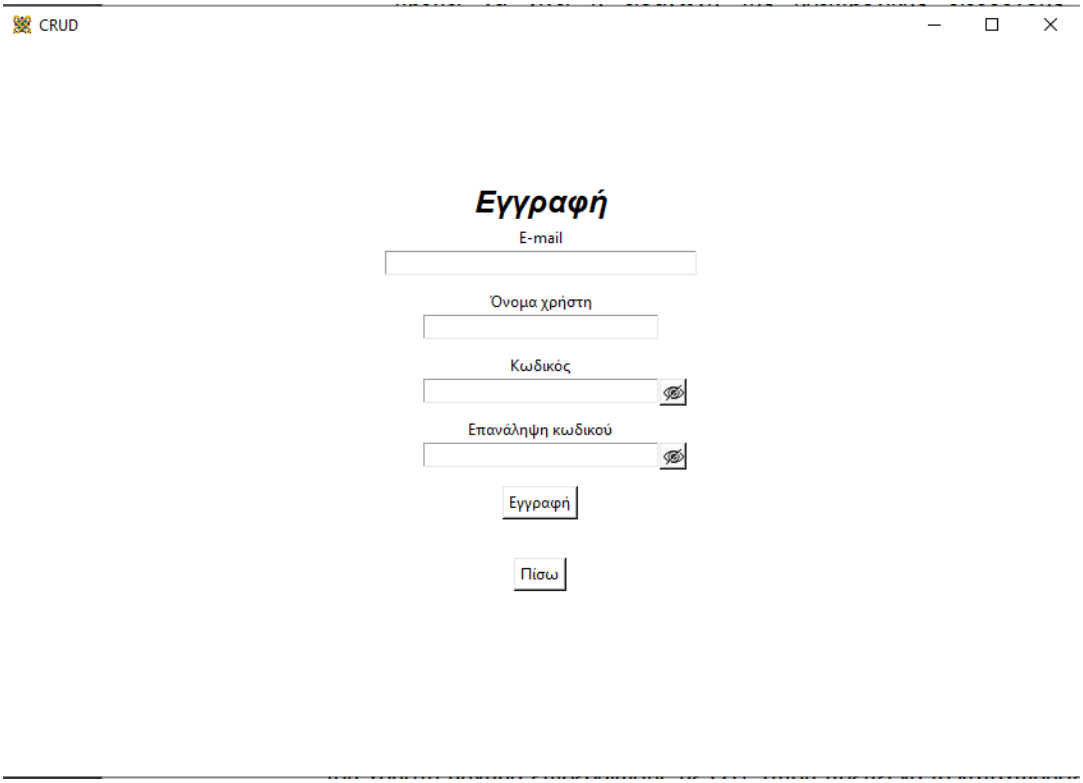
- Πληρώντας τις προϋποθέσεις κωδικού πρόσβασης εμφανίζεται μήνυμα επιτυχής αλλαγή κωδικού πρόσβασης.



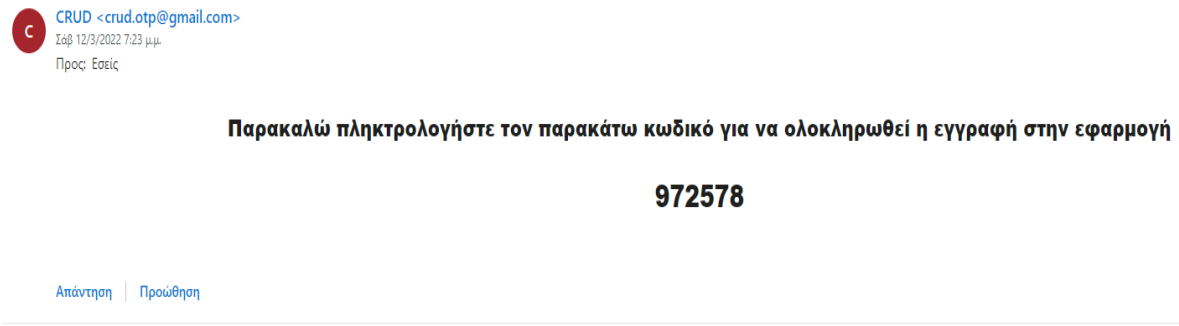
Εικόνα 43 [ΚΕΦ.3.] Μήνυμα επιτυχής αλλαγή κωδικού πρόσβασης.

Ειδάλλως σε περίπτωση που ο χρήστης δεν διαθέτει λογαριασμό, μπορεί να εγγραφεί στο σύστημα πατώντας το κουμπί εγγραφή στο αρχικό παράθυρο. Στη συνέχεια μεταβαίνει σε νέο παράθυρο όπου πρέπει να εισάγει τα προσωπικά του στοιχεία: e-mail,όνομα χρήστη, κωδικό πρόσβασης καθώς και την επαναπληκτρολόγηση του κωδικού. Αξίζει να αναφερθεί ότι πατώντας το εικονίδιο δίπλα από τον κωδικό εμφανίζονται οι χαρακτήρες που πληκτρολόγησε ο χρήστης. Τέλος πατώντας εγγραφή αποστέλλεται στην ηλεκτρονική διεύθυνση του χρήστη, μήνυμα

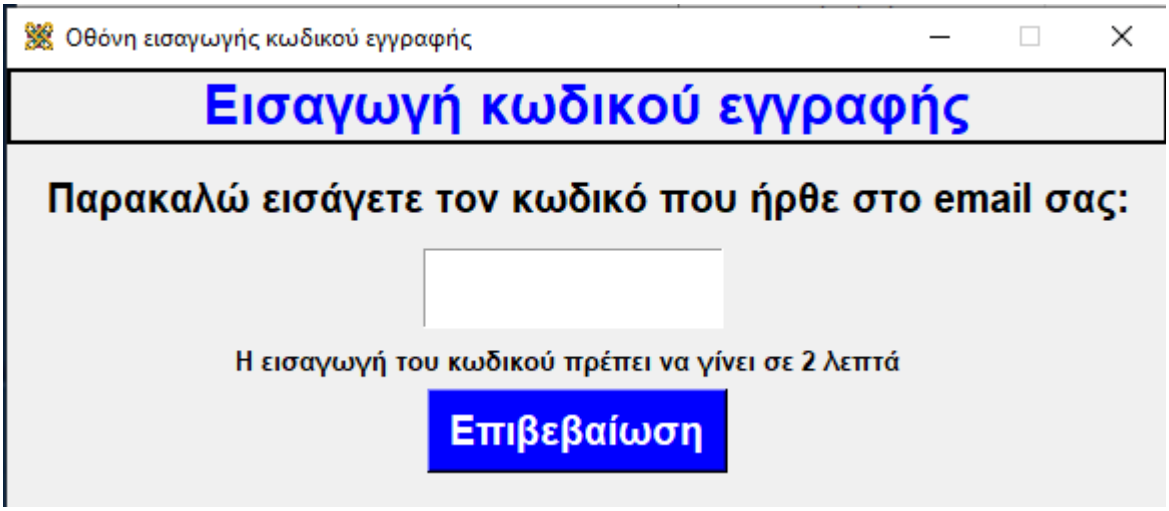
επιβεβαίωσης με OTP όπου πρέπει να το καταχωρήσει σε συγκεκριμένο χρονικό διάστημα(δύο λεπτών).



Εικόνα 44 [ΚΕΦ.3.] Παράθυρο εγγραφής χρήστη.



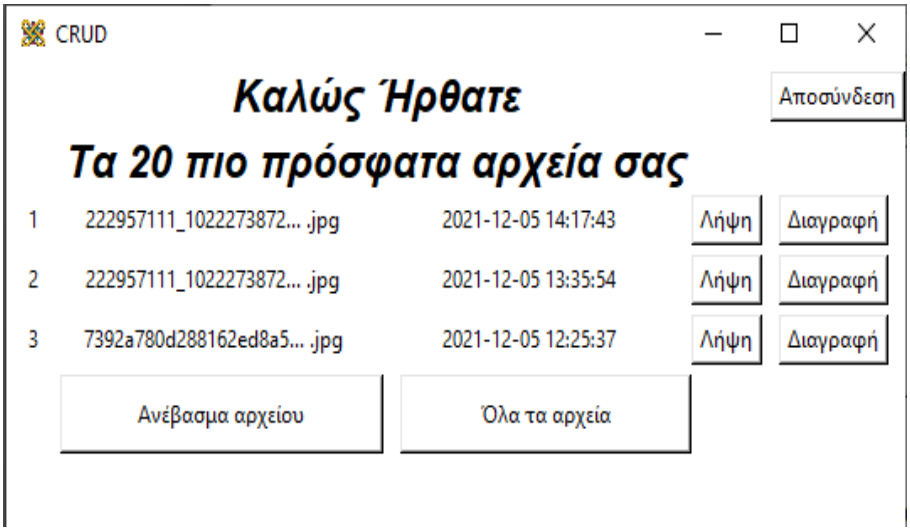
Εικόνα 45 [ΚΕΦ.3.] E-mail με OTP για ολοκλήρωση της εγγραφής.



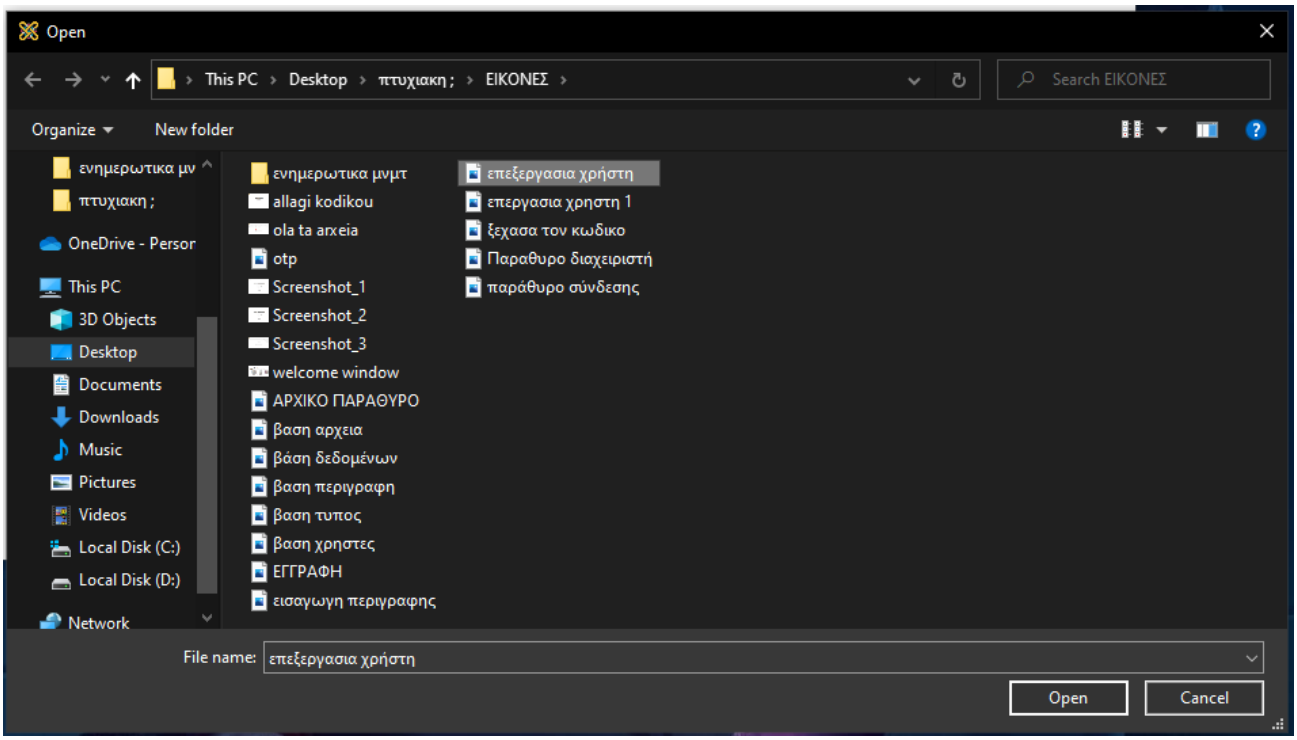
Εικόνα 46 [ΚΕΦ.3.] Εισαγωγή OTP.

3.3.1. Παράθυρο υποδοχής

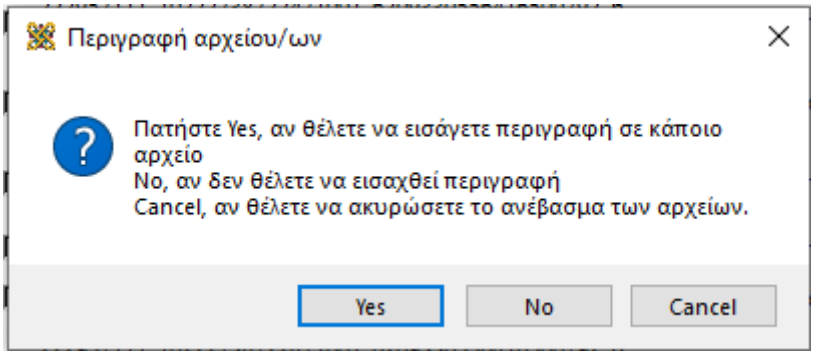
Εφόσον ο χρήστης εισάγει σωστά τα στοιχεία του ανοίγει το παράθυρο υποδοχής όπου προβάλονται τα 20 πιο πρόσφατα αρχεία του,τα οποία και έχει την δυνατότητα να τα κατέβασει ή να τα διαγράψει καθώς επίσης να ανεβάσει ένα νέο αρχείο είτε να μεταβεί στο κύριο πρόγραμμα πατώντας το κουμπί <<Ολα τα αρχεία>> είτε να αποσυνδεθεί. Πατώντας το κουμπί << Ανέβασμα αρχείου>> ο χρήστης θα πρέπει να επιλέξει κάποιο αρχείο και αν το επιθυμέι να προσθέση περιγραφή, ώστε να ανέβει στην βάση δεδομένων.



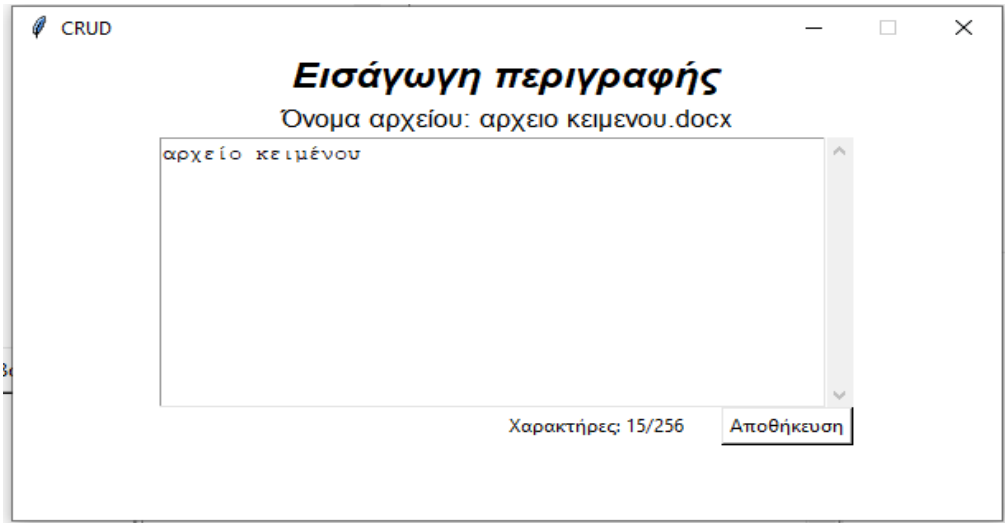
Εικόνα 47 [ΚΕΦ.3.] Παράθυρο υποδοχής χρήστη.



Εικόνα 48 [ΚΕΦ.3.] Επιλογή αρχείου για ανέβασμα.



Εικόνα 49 [ΚΕΦ.3.] [ΚΕΦ.3.] Μήνυμα περιγραφής αρχείου.



Εικόνα 50 [ΚΕΦ.3.] Παράθυρο προσθήκης περιγραφής αρχείου.

3.3.2. Κύριο παράθυρο

Μεταβαίνοντας στο κύριο παράθυρο ο χρήστης έχει την δυνατότητα να γυρίσει πίσω στο παράθυρο υποδοχής καθώς και να αποσυνδεθεί από το πρόγραμμα, έπειτα όπως παρατηρείται το παράθυρο χωρίζεται σε δύο τμήματα:

I. **Όλα τα αρχεία άλλων χρηστών:** Στο τμήμα αυτό ο χρήστης μπορεί να πραγματοποιήσει τις εξής ενέργειες:
    1. **Αναζήτηση:** Βάση το είδος των αρχείων ή πληκτρολογώντας την ονομασία του επιθυμητού αρχείου.
    2. **Επιλογή:** Επιλέγοντας ένα από αυτά γίνεται διαθέσιμη ως προς ανάγνωση η περιγραφή του τελευταίου επιλεγμένου.

3. **Λήψη:** Με αυστηρή προϋπόθεση να έχουν επιλεγεί αρχεία ο χρήστης μπορεί να τα αποθηκεύσει σε τοπικό φάκελο.

II. **Όλα τα αρχεία του χρήστη:** Όπως και στο τμήμα όλα τα αρχεία άλλων χρηστών έτσι και εδώ λειτουργεί με τον ίδιο τρόπο αλλά δίνεται στον χρήστη μια ευχέρεια παραπάνω επιλογών:

a) **Μετονομασία:** Εδώ γίνεται η μετονομασία των επιλεγμένων αρχείων του.

b) **Επεξεργασία:** Εδώ μπορεί να τροποποιήσει ή να διαγράψει την περιγραφή και στην συνέχεια να την ενημερώσει.

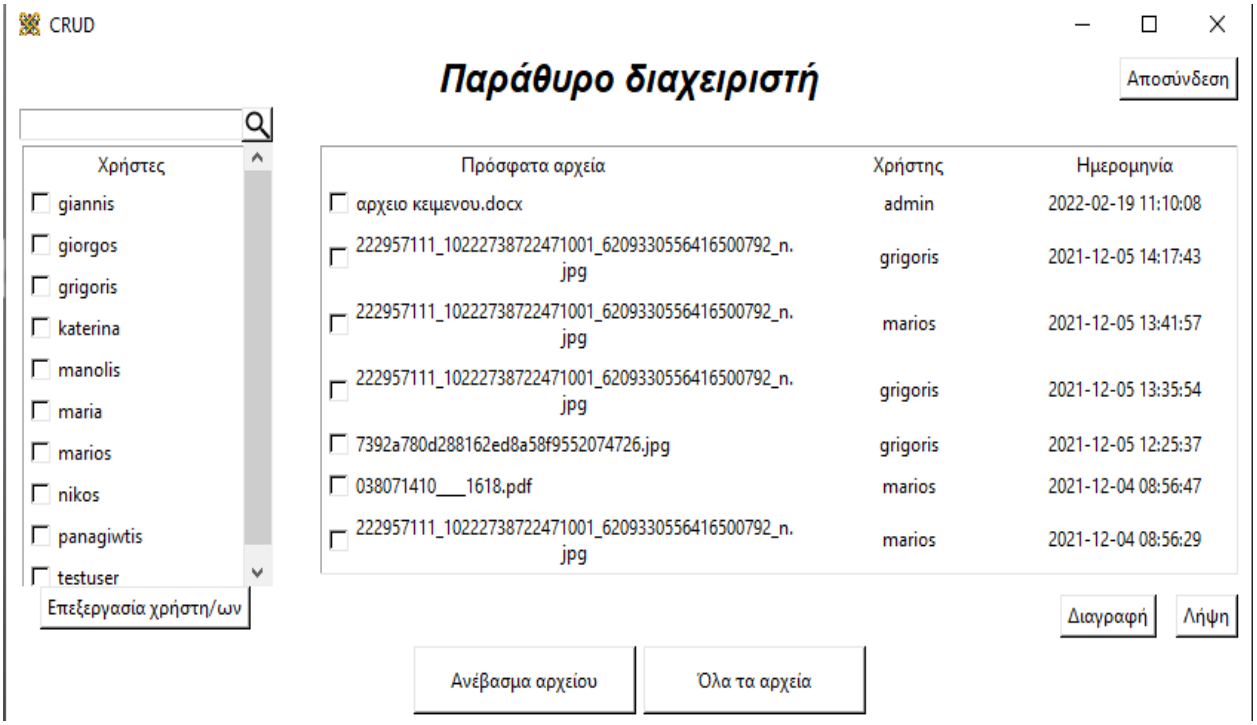
c) **Ανέβασμα:** Όπως και στο παράθυρο υποδοχής ετσι και εδω υπάρχει η δυνατότητα ανεβάσματος αρχείου με όνομα και περιγραφή στην βάση.

d) **Διαγραφή:** Στο τμήμα αυτό χρησιμοποιώντας την ενέργεια διαγραφή ο χρήστης μπορεί να διαγράψει είτε μαζικά είτε μεμονωμένα τα επιλεγμένα αρχεία.

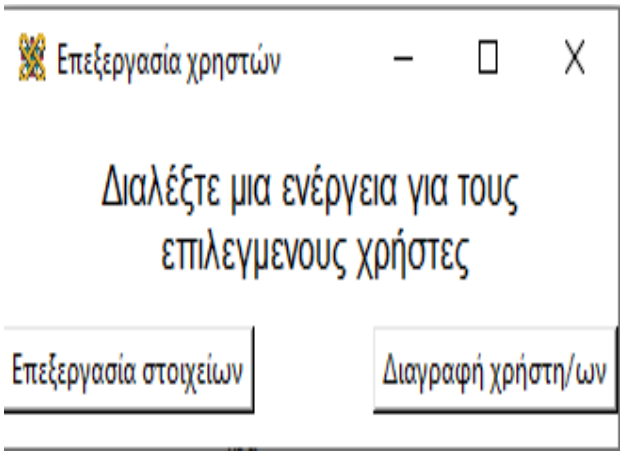
3.3.3. Παράθυρο διαχείρισης

Στο παράθυρο αυτό ο διαχειριστής μπορεί να ανεβάσει κάποιο αρχείο του καθώς και να το διαγράψει ή να κατεβάσει τα πρόσφατα αρχεία όλων των χρηστών σε αντίθεση με το παράθυρο υποδοχής του χρήστη. Παράλληλα του δίνεται η δυνατότητα να

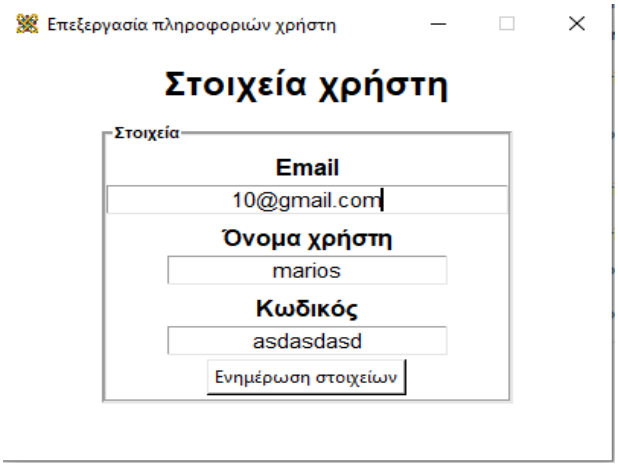
αναζητήσει και με βασική προϋπόθεση να επιλέξει έναν ή παραπάνω χρήστες πατώντας στο Επεξεργασία χρήστη/ων να μπορεί είτε να τους διαγράψει είτε επεξεργαστεί τα στοιχεία τους, μεμονωμένα ακόμα και μαζικά.



Εικόνα 51 [ΚΕΦ.3.] Παράθυρο διαχειριστή.

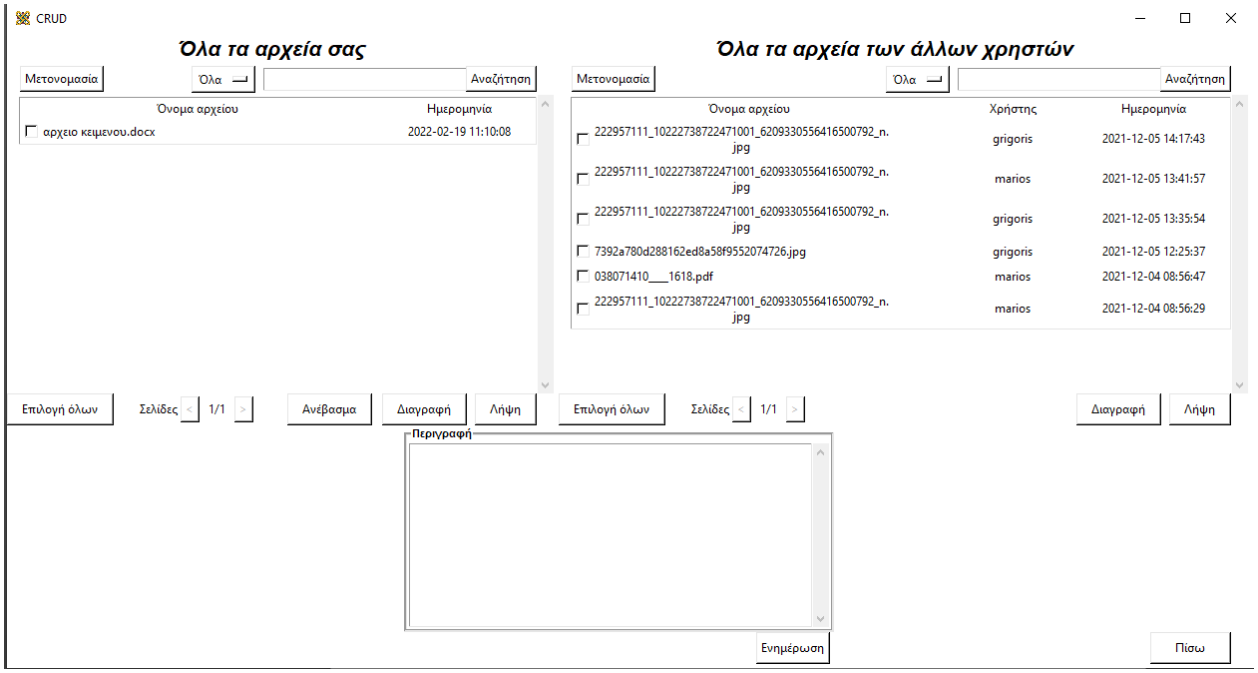


Εικόνα 52 [ΚΕΦ.3.] Παράθυρο επεξεργασίας χρηστών.



Εικόνα 53 [ΚΕΦ.3.] Παράθυρο επεξεργασίας στοιχείων.

Όπως αναφέραμε παραπάνω στο κύριο παράθυρο του χρήστη έτσι και εδώ ο διαχειριστής μπορεί να κάνει τις ίδιες ενέργειες καθώς επίσης μπορεί να επεξεργαστεί όχι μόνο τα αρχεία του αλλά όλα τα αρχεία που έχουν ανεβεί στην βάση δεδομένων.



Εικόνα 54 [ΚΕΦ.3.] Κεντρικό παράθυρο διαχειριστή.

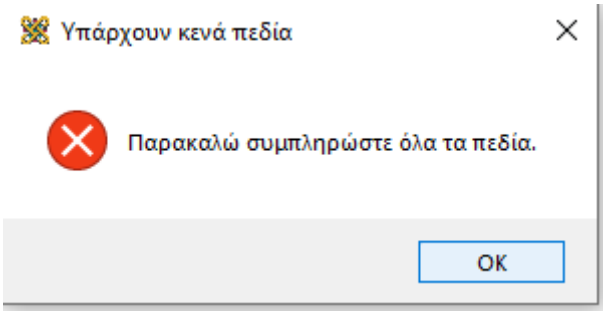
3.4. Περιπτώσεις αποτροπής λειτουργίας του προγράμματος και ενημερωρικά παράθυρα.

Το λογισμικό αυτό έχει σχεδιαστεί με σκοπό τη βέλτιστη λειτουργία και εξυπηρέτηση του χρήστη. Για την επίτευξή του , πρέπει να αποτραπούν διάφορα είδη σφαλμάτων

κάνοντας χρήση ενημερωτικών ή δια δραστικών παραθύρων που έχουν ως σκοπό την ενημέρωση του χρήστη. Τα παράθυρα που έχουν δημιουργηθεί είναι τα εξής:

3.4.1. Σφάλμα κενού πεδίου

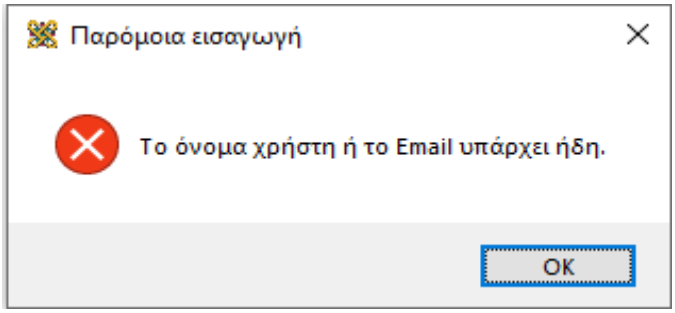
Το σφάλμα αυτό προκύπτει όταν ο χρήστης δεν έχει συμπληρώσει όλα τα απαραίτητα πεδία στο παράθυρο της σύνδεσης και της εγγραφής.



Εικόνα 55 [ΚΕΦ.3.] Μήνυμα σφάλματος κενού πεδίου.

3.4.2 Σφάλμα παρόμοια εισαγωγή.

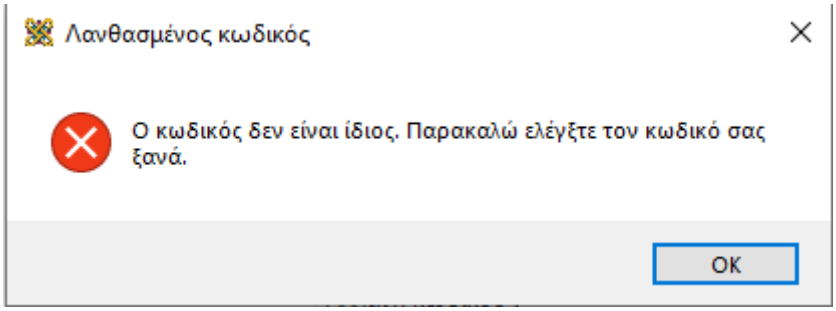
Το παρόν μήνυμα εμφανίζεται όταν το email ή το όνομα χρήστη που έχει εισαχθεί εάν υπάρχει ήδη στην βάση δεδομένων.



Εικόνα 56 [ΚΕΦ.3.] Μήνυμα σφάλματος ήδη υπάρχον e-mail.

3.4.3 Σφάλμα λανθασμένος κωδικός.

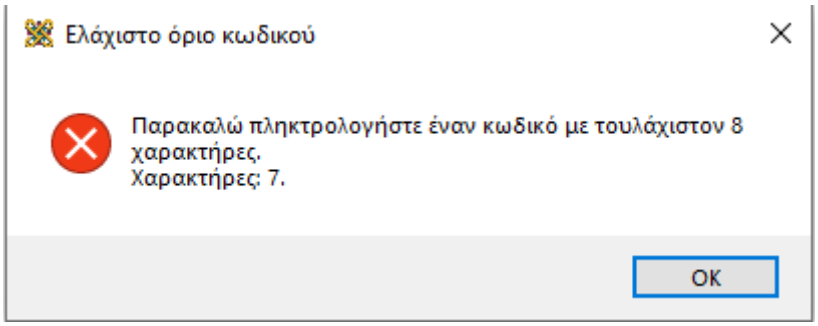
Το παρόν σφάλμα εμφανίζεται όταν ο χρήστης πληκτρολογήσει λανθασμένη επιβεβαίωση κωδικού πρόσβασης είτε στο παράθυρο εγγραφής είτε στην επιβεβαίωση στο παράθυρο ξέχασα τον κωδικό .



Εικόνα 57 [ΚΕΦ.3.] Μήνυμα σφάλματος επαλήθευσης κωδικού πρόσβασης.

3.4.4 Σφάλμα ελάχιστο όριο κωδικού πρόσβασης.

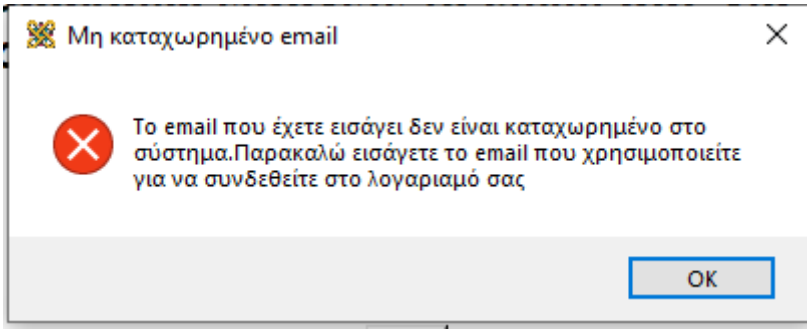
Το σφάλμα αυτό προκύπτει από εσφαλμένη πληκτρολόγηση κωδικού πρόσβασης ο οποίος είναι κάτω από το επιτρεπτό όριο αριθμών στο παράθυρο εισόδου και εγγραφής καθώς και στο παράθυρο ξέχασα τον κωδικό.



Εικόνα 58 [ΚΕΦ.3.] Μήνυμα σφάλματος ελάχιστο όριο κωδικού πρόσβασης.

3.4.5 Σφάλμα μη καταχωρημένο email.

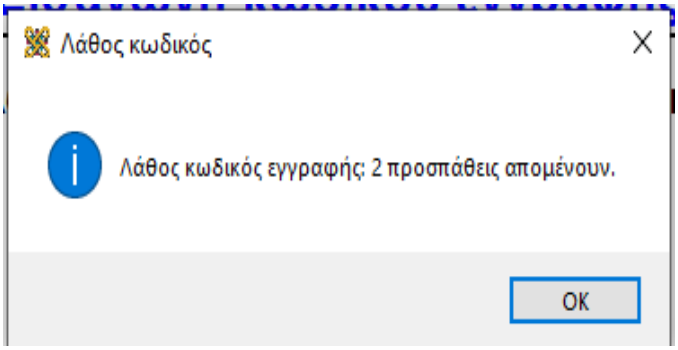
Το παρόν σφάλμα εμφανίζεται όταν το email που έχει εισαχθεί στο παράθυρο εισόδου δεν είναι καταχωρημένο στην βάση δεδομένων του συστήματος.



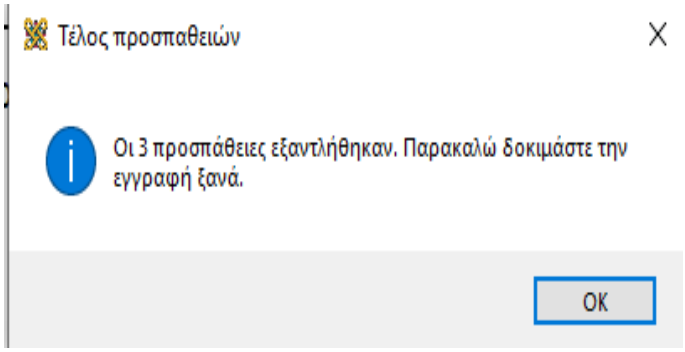
Εικόνα 59 [ΚΕΦ.3.] Μήνυμα σφάλματος μη καταχωρημένο e-mail.

3.4.6 Σφάλματα λανθασμένες προσπάθειες OTP.

Τα μηνύματα λάθους αυτά προκύπτουν από την λανθασμένη εισαγωγή του OTP μία έως τρείς φορές το σφάλμα αυτό το συναντάμε στο παράθυρο εγγραφής ή στο παράθυρο ξέχασα τον κωδικό.



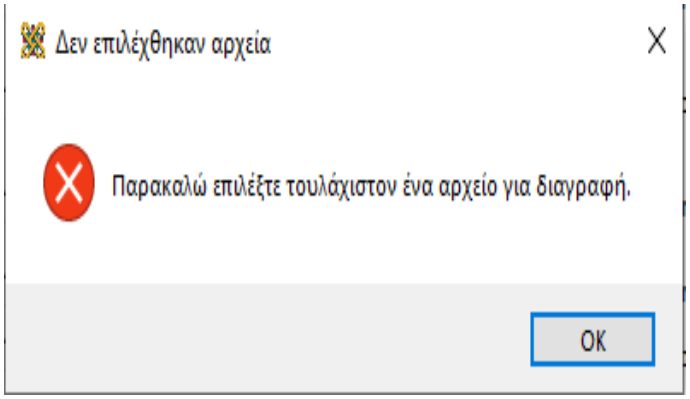
Εικόνα 60 [ΚΕΦ.3.] Μήνυμα σφάλματος λανθασμένου κωδικού.



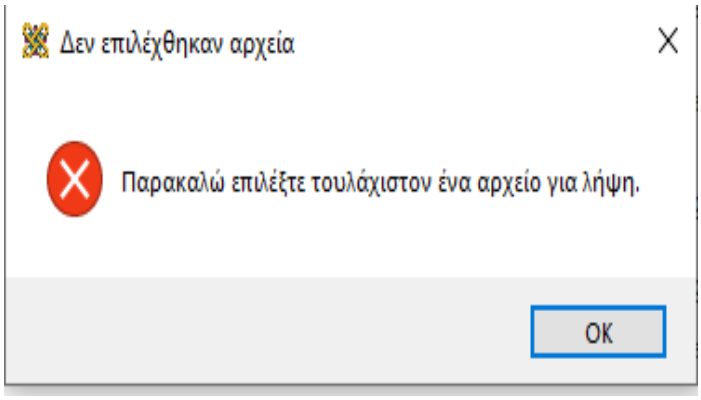
Εικόνα 61 [ΚΕΦ.3.] Μήνυμα σφάλματος τέλος προσπαθειών.

3.4.7 Σφάλματα δεν επιλέχθηκαν αρχεία.

Τα παρόν μηνύματα λάθους εμφανίζονται οταν ο χρήστης δεν έχει επιλέξει κανένα αρχείο και δοκιμάσει να κάνει λήψη ή διαγραφή αρχείου. Τα σφάλματα αυτά τα συναντάμε στα παράθυρο υποδοχής όπως και στο παράθυρο όλα τα αρχεία του χρήστη καθώς επισής και στο παράθυρο του διαχειριστή οπως και στο κύριο παράθυρό του.



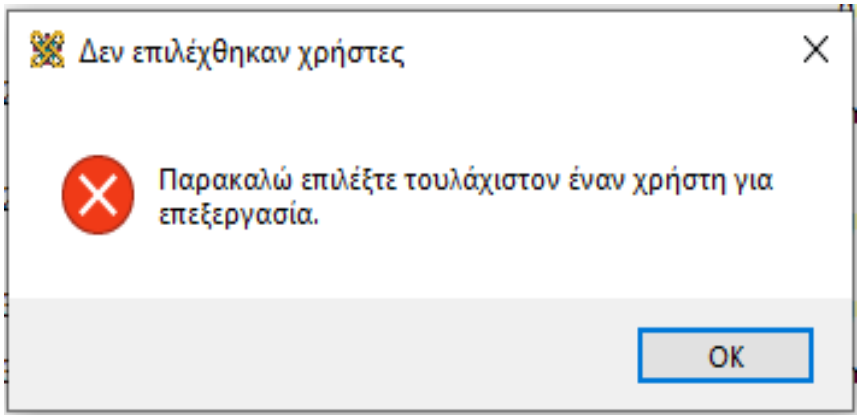
Εικόνα 62 [ΚΕΦ.3.] Μήνυμα σφάλματος μη επιλογής αρχείου προς διαγραφή.



Εικόνα 63 [ΚΕΦ.3.] Μήνυμα σφάλματος μη επιλογής αρχείου προς λήψη.

3.4.8 Σφάλμα δεν επιλέχθηκαν χρήστες.

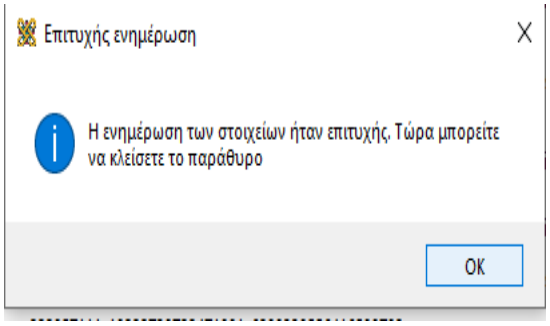
Το παρόν μήνυμα προκύπτει από την ενέργεια του κουμπιού επεξεργασία χρήστη/ων στο παράθυρο του διαχειριστή όταν δεν έχει επιλεχθεί κανένας χρήστης.



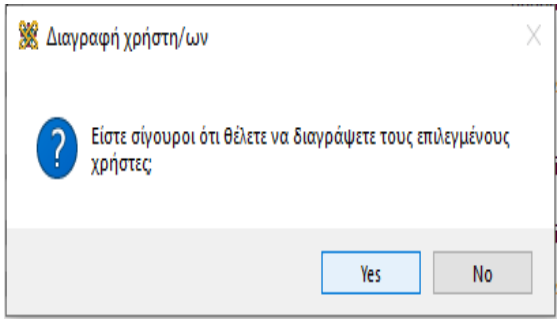
Εικόνα 64 [ΚΕΦ.3.] Μήνυμα σφάλματος μη επιλογής χρηστών.

3.4.9 Ενημερωτικά παράθυρα επεξεργασίας και διαγραφής χρήστη.

Τα παράθυρα αυτά προκύπτουν από την επεξεργασία ενός ή παραπάνω χρηστών στο παράθυρο του διαχειριστή όπου τον ενημερώνει για την επιτυχή αλλαγή στοιχείων, όπως αντίστοιχα του γίνεται ερώτηση αν είναι σίγουρος οτι θέλει να διαγράψει τους συγκεκριμένους χρήστες.



Εικόνα 65 [ΚΕΦ.3.] Μήνυμα επιτυχής ενημέρωσης.



Εικόνα 66 [ΚΕΦ.3.] Μήνυμα ερώτησης διαγραφής χρήστη.

Συμπεράσματα

- Πολλοί χρήστες χρησιμοποιούν πλέον προγράμματα διαχείρισης αρχείων προς την διευκόλυνσή τους. Η χρήση τους επιτρέπει την καλύτερη διαχείριση και πρόσβαση αρχείων και σημειώσεων μεταξύ των χρηστών αυτό έχει ως αποτέλεσμα την μείωση του χρόνου διαμοιρασμού των πληροφοριών.
- Υπάρχουν πολλές εφαρμογές διαχείρισης αρχείων στο διαδίκτυο που θα μπορούσε μια επιχείρηση ή μια ομάδα ατόμων να χρησιμοποιήσει είτε δωρεάν είτε επι πληρωμή. Συγχρόνως, ο λόγος για την απόκτηση μια τέτοιας εφαρμογής ποικίλει σε λόγω της εύκολης προσαρμογής στις απαιτήσεις του χρήστη. Μπορεί να χρησιμοποιηθεί από επιχειρίσεις μέχρι απλούς χρήστες όπου θέλουν άμεση πρόσβαση στις πληροφορίες αυτες από οπουδήποτε και οποιονδήποτε τις χρειαστεί.
- Δημιουργώντας την εφαρμογή αυτή, παρατηρήθηκε ότι η ανάπτυξη ενός τέτοιου περιβάλλοντος πρέπει να είναι κατανοητό και εύχρηστο για την καλύτερη εξυπηρέτηση και την εμπειρία του χρήστη. Κύριο προτέρημα της εφαρμογής CRUD σε σχέση με τις υφιστάμενες εφαρμογές είναι ότι η βάση δεδομένων μπορεί να εγκατασταθεί σε οποιονδήποτε server που επιθυμεί η εταρεία που την χρησιμοποιεί.
- Μελλοντικές Βελτιώσεις στο παρόν είναι χρήση άλλων τεχνολογιών και μεθόδων για την επίτευξη μοντέρνου σχεδιασμού φιλικό και ευπαρουσίαστο προς τον χρήστη, δημιουργία δωματίων τηλεδιάσκεψης μεταξύ των χρηστών της εφαρμογής, δημιουργίας και ανταλλαγής σημειώσεων , δυνατότητα ανεβάσματος αρχείου με προστασία κωδικού, πρόσβαση αρχείων με βάση των δικαιωμάτων του χρήστη καθώς επίσης δημιουργία σύστημα κρυπτογράφισης αρχείων κατά την καταχώρηση στην βάση δεδομένων.

Βιβλιογραφία

Ηλεκτρονικές Πηγές

ArgoUML. (2023). Ανάκτηση από https://argouml.en.softonic.com

AutoPyToExe. (2023). Ανάκτηση από https://pypi.org/project/auto-py-to-exe/

Box. (2023). Ανάκτηση από https://www.box.com/

Citrix ShareFile. (2023). Ανάκτηση από https://www.sharefile.com/

Inno Setup. (2020). Ανάκτηση από https://jrsoftware.org/isinfo.php

PyCharm. (2023). Ανάκτηση από https://www.jetbrains.com/pycharm/

Python. (2023). Ανάκτηση από https://www.python.org/

Tresorit. (2023). Ανάκτηση από https://tresorit.com/

Wikipedia. (2023). Ανάκτηση από https://en.wikipedia.org/wiki/File_sharing

Xampp. (2023). Ανάκτηση από https://www.apachefriends.org/

Ελληνικά Βιβλία

Βασίλης, Γ. (2006). *Αντικειμενοστραφής ανάπτυξη λογισμικού με τη UML. 1η Έκδοση* . Κλειδάριθμος.

Ξένα βιβλία

B. Lubanovic.(2019). Introducing Python.

DATE, C. J. (1998). *Εισαγωγή στα συστήματα βάσεων δεδομένων* (6η Έκδοση εκδ.). Αθήνα: Ιωάννης Φαλδαμής.

Dennis, A. (2010). *Ανάλυση & σχεδιασμός συστημάτων με τη UML.* Αθήνα: Κλειδάριθμος.

Παράρτημα κώδικα της εφαρμογής CRUD

CRUD_main.py

```
import re
import socket
import subprocess
import sys
import tkinter as tk  # installed through pip install command
from tkinter import font as tkfont  # python 3
from tkinter import StringVar
from tkinter import OptionMenu
from tkinter import messagebox
from tkinter import filedialog
from tkinter import scrolledtext
from email.mime.text import MIMEText  # installed through pip install
command
from email.mime.multipart import MIMEMultipart  # installed through
pip install command
import concurrent.futures  # python embedded library
from email.header import Header  # python embedded library
from email.utils import formataddr  # python embedded library
from tkinter import ttk  # installed through pip install command
from PIL import Image, ImageTk  # installed through pip install
command
import os  # python embedded library
import math  # python embedded library
import random  # python embedded library
import datetime  # python embedded library
from functools import partial  # installed through pip install
command
from threading import Thread  # python embedded library
import mysql.connector  # installed through pip install command
import smtplib  # installed through pip install command

regex = re.compile(r"(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-
.]+[a-zA-Z0-9-.]+$)")

image_extensions = ["ase", "art", "bmp", "blp", "cd5", "cit", "cpt",
"cr2", "cut", "dds", "dib", "djvu", "egt", "exif",
                    "gif", "gpl", "grf", "icns", "ico", "iff", "jng",
"jpeg", "jpg", "jfif", "jp2", "jps", "lbm", "max",
                    "miff", "mng", "msp", "nitf", "ota", "pbm",
"pc1", "pc2", "pc3", "pcf", "pcx", "pdn", "pgm", "PI1",
                    "PI2", "PI3", "pict", "pct", "pnm", "pns", "ppm",
"psb", "psd", "pdd", "psp", "px", "pxm", "pxr",
                    "qfx", "raw", "rle", "sct", "sgi", "rgb", "int",
"bw", "tga", "tiff", "tif", "vtf", "xbm", "xcf",
                    "xpm", "3dv", "amf", "ai", "awg", "cgm", "cdr",
"cmx", "dxf", "e2d", "egt", "eps", "fs", "gbr",
                    "odg", "svg", "stl", "vrml", "x3d", "sxd", "v2d",
"vnd", "wmf", "emf", "art", "xar", "png", "webp",
                    "jxr", "hdp", "wdp", "cur", "ecw", "iff", "lbm",
"liff", "nrrd", "pam", "pcx", "pgf", "sgi", "rgb",
                    "rgba", "bw", "int", "inta", "sid", "ras", "sun",
"tga", "nef", "crw"]
```

```
text_extensions = ["pdf", "sty", "fdx", "lis", "lst", "sub", "log",
"err", "mbox", "txt", "docx", "wpd", "lst", "dwd", "wpt", "bad",
"bdr",
                   "btd", "dotm", "dvi", "fadein", "fdr", "flr",
"kes", "klg", "kon", "lxfml", "mellel", "mnt", "mwd", "mwp",
                   "safetext", "apt", "asc", "dropbox", "wri", "aim",
"diz", "fbl", "vnt", "gdoc", "ans", "gpd", "xwp", "me", "upd",
                   "rtf", "text", "hht", "prt", "run", "saf", "doc",
"doc", "docm", "emlx", "nfo", "ott", "qdl", "tab", "xdl", "epp",
                   "jp1", "save", "prt", "ris", "wps", "abw", "sig",
"bdp", "dx", "fpt", "frt", "hs", "pwd", "rtfd", "template",
                   "dsv", "eit", "fadeintemplate", "pfs", "tpc",
"msg", "1st", "tmd", "dotx", "lp2", "ltx", "pages", "eml", "odm",
                   "se", "tab", "man", "wps", "odt", "bib", "cod",
"wpl", "tex", "rst", "ase", "aww", "bean", "bib", "bna", "boc",
                   "bzabw", "calca", "charset", "chord", "crwl",
"cyi", "dne", "eio", "etf", "fcf", "fdt", "fdxt", "fodt", "fountain",
                   "fwd", "gsd", "gthr", "gv", "ipspot", "klg",
"knt", "kwd", "lbt", "luf", "lyx", "md5txt", "mell", "nb", "njx",
                   "nwp", "ofl", "pages-tef", "pjt", "plantuml",
"psw", "pu", "pwi", "qpf", "readme", "rpt", "rzk", "rzn", "scriv",
                   "scrivx", "sct", "scw", "sgm", "slagz", "story",
"strings", "sublime-project", "sublime-workspace", "tdf", "tdf",
                   "textclipping", "u3i", "unauth", "unx", "utf8",
"utxt", "wpa", "wpd", "xbdoc", "xdl", "xy3", "xyp", "xyw", "zabw",
                   "zrtf", "apkg", "ascii", "docxml", "jarvis",
"latex", "lyt", "now", "openbsd", "wtx", "xy", "sla", "idx", "asc",
                   "cnm", "emulecollection", "hwp", "xwp", "hbk",
"odo", "p7s", "notes", "rtd", "sdw", "sms", "ssa", "sxw", "sam",
                   "wp6", "scc", "lnt", "lwp", "uot", "vct",
"webdoc", "act", "aty", "awp", "awt", "bbs", "bml", "brx", "chart",
                   "cws", "dgs", "dxb", "dxp", "err", "fdf", "fds",
"gpn", "jis", "min", "mw", "ndoc", "ngloss", "nwctxt", "nwm",
                   "ocr", "ort", "pdpcmd", "pfx", "pmo", "pvj",
"pvm", "pwdp", "pwdpl", "pwr", "rtx", "sam", "scm", "sdm", "session",
                   "skcard", "smf", "sxg", "tlb", "tm", "tmv",
"trelby", "tvj", "uof", "wbk", "wp", "wp4", "wp5", "wp7", "wpt",
                   "xbplate", "hwp", "wpd", "zw", "emf", "gmd", "hz",
"stw", "xwp", "dca", "docz", "dox", "dsc", "etx", "euc", "faq",
                   "fft", "fwdn", "iil", "ipf", "joe", "jrtf", "ltr",
"lue", "mcw", "odif", "rad", "rft", "sdoc", "thp", "vw", "wn",
                   "wpw", "wsd"]

video_extensions = ["webm", "mkv", "flv", "vob", "ogv", "ogg", "rrc",
"gifv", "mng", "mov", "avi", "qt", "wmv", "yuv", "rm", "asf",
                    "amv", "mp4", "m4p", "m4v", "mpg", "mp2", "mpeg",
"mpe", "mpv", "m4v", "svi", "3gp", "3g2", "mxf", "roq", "nsv",
                    "flv", "f4v", "f4p", "f4a", "f4b"]

sound_extensions = ["caf", "abm", "oga", "omf", "pla", "asd", "bnk",
"bun", "csh    C", "hsb", "mscz", "nsf", "rfl", "sma", "sng",
                    "vag", "acm", "dss", "aob", "pkf", "syx", "ksd",
"saf", "m4r", "ang", "sf2", "mid", "xfs", "mod", "act", "vdj",
                    "sfk", "als", "mp3", "wma", "wav", "ins", "gp5",
"sty", "afc", "amxd", "at3", "dmsa", "dmse", "emp ",
                    "ftm", "mxl", "nbs", "nra", "pcast", "ptx",
"rns", "slp", "trak", "a2b", "a2i", "agr", "akp", "bww", "copy",
                    "dfc", "dls", "drg", "dsf", "dsm", "dtm", "flp",
"frg", "isma", "krz", "mbr", "minigsf", "mtp", "musx", "nkm",
                    "omg", "peak", "rex", "rip", "rol", "sfpack",
"smf", "smp", "sseq", "svd", "syh", "syw", "tg ", "u", "uax", "vmd",
```

```
                    "vpl", "vyf", "wve", "zvd", "669", "aimppl",
"eop", "nvf", "midi", "sib", "flac", "aup", "vlc", "xspf", "aif",
                    "m3u", "zab", "seq", "mpga", "pcg", "wpk",
"logic", "ogg", "ram", "rx2", "5xe", "fev", "sdat", "mus", "flp",
"xa",
                    "mui", "aac", "nki", "dig", "gsm", "nsa", "wave",
"dff", "pcm", "pho", "q1", "rmf", "u8", "ym", "m4a", "cpr",
                    "ac3", "aiff", "ins", "oma", "sds", "3ga", "amr",
"sesx", "acd-zip", "4mp", "apl", "cwp", "cwt", "ds2", "gpk",
                    "gsflib", "med", "mx5", "ply", "qcp", "r1m",
"rmj", "stm", "w64", "au", "b4s", "ds", "hbb", "ins", "it", "kit",
                    "mdl", "mu3", "q2", "sbg", "sfap0", "vgz", "vmf",
"zpa", "2sf", "cda", "m3u8", "aa", "fls", "pd", "mus", "mpa",
                    "emx", "pls", "adg", "mmm", "sd", "vox", "m4b",
"ape", "mus", "xm", "snd", "nwc", "wpp", "vb", "iff", "ra",
                    "acd", "amz", "dcf", "5xb", "5xs", "a2m", "abc",
"acd-bak", "adts", "agm", "aifc", "alc", "amf", "au", "aud",
                    "band", "bap", "bdd", "bidule", "bwf", "caff",
"cdda", "cdlx", "cdo", "cel", "cgrp", "cidb", "ckb", "conform",
                    "cpt", "cts", "cwb", "cws", "dct", "dewf", "df2",
"dig", "dm", "dmf", "dra", "dtshd", "dwd", "efk", "efq", "efs",
                    "efv", "emd", "esps", "f2r", "f32", "f3r", "f4a",
"f64", "fdp", "flm", "fsb", "fsc", "fsm", "ftm", "ftmx", "fzf",
                    "fzv", "g721", "g726", "gig", "gpbank", "groove",
"gsf", "h4b", "h5b", "h5s", "hbe", "igp", "iti", "kfn", "koz",
                    "koz", "ksf", "kt3", "la", "lso", "lwv", "m4p",
"ma1", "mdc", "mgv", "miniusf", "mmp", "mmpz", "mo3", "mpc",
                    "mte", "mtf", "mti", "mtm", "mux", "narrative",
"ncw", "nkb", "nkc", "nks", "nkx", "nml", "note", "nrt", "nst",
                    "ntn", "obw", "okt", "omx", "ots", "ovw",
"pandora", "pca", "pek", "pna", "psm", "ptm", "pts", "rax", "rgrp",
                    "rmi", "rmx", "rng", "rsn", "rso", "rti", "s3i",
"s3m", "sbi", "sc2", "scs11", "sd", "sd2", "sfz", "sgp", "smpx",
                    "sou", "sppack", "sprg", "stap", "sty", "sxt",
"syn", "td0", "tta", "txw", "ult", "uni", "usf", "usflib", "ust",
                    "uw", "uwf", "vap", "vc3", "vmo", "voc", "voxal",
"vpw", "vrf", "vsq", "wfb", "wfd", "wfm", "wfp", "wow", "wproj",
                    "wrk", "wus", "wut", "wv", "wvc", "wwu", "xmu",
"xrns", "yookoo", "adv", "gmc", "mp_", "ppcx", "sns", "vce",
                    "vgm", "xmf", "xwb", "2sflib", "6cm", "8med",
"a52", "al", "d01", "dsp", "gsm", "mini2sf", "prg", "tun", "wyz",
                    "xt", "kar", "sfl", "gpx", "dts", "wax", "ove",
"ddt", "sf", "cdr", "dvf", "vpm", "opus", "sid", "aa3", "ptf",
                    "adt", "efa", "fpa", "gbs", "h5e", "mpdp", "odm",
"pk", "slx", "stx", "swa", "vqf", "vsqx", "w01", "zpl", "bmml",
                    "brstm", "mmp", "mscx", "ppc", "rsf", "sdt",
"xa", "xpf", "rtm", "wem", "tak", "g723", "rbs", "sap", "ear", "wav",
                    "pat", "ams", "ics", "k26", "mka", "mmf", "mp2",
"mts", "myr", "psf", "ses", "shn", "snd", "a2p", "a2t", "a2w",
                    "ab", "acp", "ahx", "ais", "alaw", "all", "apf",
"aria", "ariax", "axa", "bwg", "c01", "ckf", "cmf", "djr", "dmc",
                    "efe", "emy", "erb", "far", "gbproj", "gym",
"h0", "h3b", "h3e", "h4e", "hbs", "hdp", "hma", "hps", "iaa", "igr",
                    "imp", "itls", "its", "jam", "jam", "kmp", "kpl",
"ksc", "kt2", "l", "lof", "lqt", "m", "m1a", "m2", "minipsf",
                    "minipsf2", "mogg", "mpu", "mt2", "mux", "mx3",
"mx4", "mx5template", "mzp", "npl", "ofr", "ovw", "pac", "pbf",
                    "phy", "pjunoxl", "plst", "pno", "prg", "psf1",
"psf2", "psy", "ptcop", "pvc", "rad", "raw", "rbs", "rcy", "rmm",
                    "rta", "rts", "rvx", "s3z", "sbk", "sd2f", "smp",
"sng", "sph", "spx", "sseq", "ssnd", "svq", "svx", "thx", "toc",
```

```
                    "tsp", "ub", "ulaw", "v2m", "vmf", "vtx", "wtpl",
"wtpt", "xbmml", "xmi", "xmz", "xsb", "xsp", "zgr", "box",
                    "imf", "sb", "sdx", "aax", "mp1", "msv", "d00",
"8svx", "ams", "dcm", "mxmf", "nmsv", "xi", "zvr", "ase", "awb",
                    "dw", "expressionmap", "mlp", "sfs", "snd",
"tak", "8cm", "gm", "k25", "cfa", "ay", "lvp", "alac", "atrac",
"avr",
                    "bcs", "bonk", "cfxr", "dwa", "evr", "fda",
"fff", "fzb", "gio", "gio", "gro", "hmi", "jo", "jo-7z", "kin",
"ksm",
                    "ktp", "minincsf", "mt9", "musa", "muz", "mwand",
"mws", "nap", "orc", "pmpl", "r", "record", "sam", "sdii", "seg",
                    "snsf", "sth", "sti", "stw", "sw", "swav", "syn
", "tfmx", "tm2", "tm8", "tmc", "ulw", "val", "voi", "wand", "xp"]

login_username = ""
login_password = ""
register_name = ""
register_email = ""
register_password = ""
register_password_ver = ""
row = ""


class SampleApp(tk.Tk):
    # Initiates a parent frame for the frames to be stacked
    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)

        self.title_font = tkfont.Font(family='Helvetica', size=18,
weight="bold", slant="italic")

        # the container is where we'll stack a bunch of frames
        # on top of each other, then the one we want visible
        # will be raised above the others
        container = tk.Frame(self)
        container.pack(side="top", expand=True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}
        # Create the Frames
        for F in (AuthenticationForm, LoginForm, RegisterForm,
HomePage, ShowAllFiles, AdminPage, ForgetPass, ChangeCode):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("AuthenticationForm")

    def show_frame(self, page_name):
        for frame in self.frames.values():
            frame.grid_remove()
        frame = self.frames[page_name]
        frame.grid()

    def destroy_frame(self):
        os.execv("CRUD_main.exe", sys.argv)
```

```
# Frame of user showing the 20 recent files he uploaded
class HomePage(tk.Frame):
    # this method returns the 20 most recent files of the user
    @staticmethod
    def get_recent_files():
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cur = cnx.cursor()
        try:
            query = "SELECT  files.id , files.file_name ,
files.date_added , users.name, file_types.name " \
                    "FROM files INNER JOIN file_types ON
files.file_type_id = file_types.id " \
                    f"INNER JOIN users ON users.id = %s AND
files.user_id = %s " \
                    "ORDER BY files.date_added DESC, files.id DESC
LIMIT 20"
            cur.execute(query, (int(row[0]), int(row[0])))
            rows = cur.fetchall()
            return rows
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Calling function readBLOB() for each specific download button
created
    def assign_buttons(self, n):
        self.readBLOB(int(row[0]), button_identities[n][1])

    # Calling function delete_file() for each delete button created
    def delete_buttons(self, n):
        controller = self.controller
        self.delete_file(delete_button_identities[n][1], controller)

    # Initiates the HomePage Frame
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.error_list = []
        self.controller = controller
        label = tk.Label(self, text="Καλώς Ήρθατε",
font=self.controller.title_font)
        label.grid(row=0, column=1, columnspan=2, sticky="nsew")
        label_recent = tk.Label(self, text="Τα 20 πιο πρόσφατα αρχεία
σας", font=self.controller.title_font)
        label_recent.grid(row=1, column=1, columnspan=2,
sticky="nsew")
        log_out = tk.Button(self, text="Αποσύνδεση")
        log_out.config(command=lambda: [self.clear_buttons(),
self.clear_labels(), text_clear(),
self.controller.show_frame("AuthenticationForm"),
self.destroy_all_frames(),
```

```
                                        root.geometry("%dx%d+%d+%d" %
(width, height, app.winfo_screenwidth() / 2 - width / 2,

app.winfo_screenheight() / 2 - height / 2))])
        log_out.grid(row=0, column=1, columnspan=4, sticky="e")

    # Creates the checkboxes, labels and buttons shown in HomePage
    def change_values(self, controller):
        global upload_button
        global delete_button
        global label0
        global label1
        global label2
        global label4
        global button_identities
        global delete_button_identities

        counter = 2
        count = 1
        button_count = 0
        button_identities = []
        delete_button_identities = []
        self.label_identities = []
        for rows in self.get_recent_files():
            label_text = rows
            if len(label_text[1]) > 20:
                file_name = label_text[1].split('.')[0][:20] + "... "
+ label_text[1][label_text[1].rindex('.'):]
                label1 = tk.Label(self, text=file_name)
            else:
                label1 = tk.Label(self, text=label_text[1])
            label2 = tk.Label(self, text=label_text[2])
            label4 = tk.Button(self, text="Λήψη",
command=partial(self.assign_buttons, button_count))
            delete_button = tk.Button(self, text="Διαγραφή",
command=partial(self.delete_buttons, button_count))
            label0 = tk.Label(self, text=str(count))
            button_identities.append((label4, label_text[0]))
            delete_button_identities.append((delete_button, rows[0]))
            self.label_identities.append((label0, label1, label2))

            label0.grid(row=counter, column=0, sticky="nsew",
padx=(10, 10), pady=(0, 5))
            label1.grid(row=counter, column=1, sticky="nsew",
padx=(0, 10), pady=(0, 5))
            label2.grid(row=counter, column=2, sticky="nsew",
padx=(0, 10), pady=(0, 5))
            label4.grid(row=counter, column=3, padx=(0, 10), pady=(0,
5), sticky="nsew")
            delete_button.grid(row=counter, column=4, padx=(0, 10),
pady=(0, 5), sticky="nsew")

            counter = counter + 1
            count += 1
            button_count += 1

        if self.get_recent_files() == []:
            upload_button = tk.Button(self, text="Ανέβασμα αρχείου",
width=20, height=2,
                                      command=lambda:
[self.clear_buttons(), self.upload(), self.check_error_list()])
```

```
        else:
            upload_button = tk.Button(self, text="Ανέβασμα αρχείου",
width=20, height=2,
                                      command=lambda:
[self.clear_buttons(), self.clear_labels(), self.upload(),
self.check_error_list()])
        self.show_all_button = tk.Button(self, text="Όλα τα αρχεία",
width=20, height=2)
        width_h1 = root.winfo_screenwidth() / 2 - 1250 / 2
        height_h1 = root.winfo_screenheight() / 2 - 635 / 2

        def load_files_for_user():
            result =
controller.frames['ShowAllFiles'].show_all_user()
            return result
        self.show_all_button.config(command=lambda:
[self.controller.show_frame("ShowAllFiles"),

controller.frames['ShowAllFiles'].currentType.set('Όλα'),

controller.frames['ShowAllFiles'].currentType_user.set('Όλα'),

root.geometry("1250x635+%s+%s" % (int(width_h1), int(height_h1))),

controller.frames['ShowAllFiles'].page_assigning_user(load_files_for_
user()),

controller.frames['ShowAllFiles'].user_page_fill(load_files_for_user(
)),

controller.frames['ShowAllFiles'].page_assigning(controller.frames['S
howAllFiles'].show_all()),

controller.frames['ShowAllFiles'].page_fill(controller.frames['ShowAl
lFiles'].show_all())])

        upload_button.grid(row=counter + 1, column=1, columnspan=1,
sticky="nsew")
        self.show_all_button.grid(row=counter + 1, column=2,
columnspan=1, sticky="nsew", padx=(10, 0))
        for i in range(counter + 1):
            self.grid_rowconfigure(i, weight=1)
        self.grid_columnconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure(2, weight=1)
        if counter >= 20:
            width_h = root.winfo_screenwidth() / 2 - 500 / 2
            height_h = root.winfo_screenheight() / 2 - 720 / 2
            root.geometry("600x700+%s+%s" % (int(width_h),
int(height_h)))
        else:
            self.resize()

    # Creates error message for the files that couldn't be uploaded
    def check_error_list(self):
        if len(self.error_list):
            files = []
            for name in self.error_list:
                files.append(name.rsplit('/', 1)[-1])
            messagebox.showerror("Το αρχείο είναι πολύ μεγάλο",
                                 "Το αρχείο που προσπαθήσατε να
```

```
ανεβάσετε είναι μεγαλύτερο από το μέγιστο όριο."
                                " \nΑρχείο: %s" % ', '.join(files))
        self.error_list = []

    # Destroys all buttons on the HomePage
    def clear_buttons(self):
        upload_button.destroy()
        self.show_all_button.destroy()
        if len(delete_button_identities):
            for n, c in delete_button_identities:
                n.destroy()
        if len(button_identities):
            for n, c in button_identities:
                n.destroy()
        self.grid_forget()
        self.resize()

    def destroy_all_frames(self):
        self.controller.destroy_frame()

    # Destroys all the labels in HomePage
    def clear_labels(self):
        if len(self.label_identities):
            for a, b, c in self.label_identities:
                a.destroy()
                b.destroy()
                c.destroy()
        self.grid_forget()
        self.resize()

    # Converts the given file into binary data in order to
    # be uploaded in the Database
    def convertToBinaryData(self, filename):
        if os.path.getsize(filename) < 1048576:  # 16777216:
            with open(filename, 'rb') as file:
                binaryData = file.read()
            return binaryData
        else:
            self.error_list.append(filename)

    # Uploads the file to the Database
    def insertBLOB(self, filename, file, user_id, filetype_id,
date_added):
        print("Inserting BLOB into files table")
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            sql_insert_blob_query = "INSERT INTO files (file_name,
file, user_id, file_type_id, date_added) " \
                                    "VALUES (%s,%s,%s,%s,%s)"

            up_file = self.convertToBinaryData(file)

            # Convert data into tuple format
            insert_blob_tuple = (filename, up_file, user_id,
```

```
filetype_id, date_added)
            cursor.execute(sql_insert_blob_query, insert_blob_tuple)
            cnx.commit()
            print("File inserted successfully as a BLOB into files
table")
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Choose the files to be uploaded
    def upload(self):
        file_path = filedialog.askopenfilenames()
        self.files = list()
        self.description = list()
        for file in file_path:
            self.files.append(file)
        if len(self.files):
            question = messagebox.askyesnocancel("Περιγραφή
αρχείου/ων", "Πατήστε Yes, αν θέλετε να εισάγετε περιγραφή σε κάποιο
αρχείο\n"

"No, αν δεν θέλετε να εισαχθεί περιγραφή\nCancel, αν θέλετε να
ακυρώσετε το ανέβασμα"

" των αρχείων.")
            if question is True:
                for f in self.files:
                    self.description_pop(filename=f.rsplit('/', 1)[-
1])
                    root.wait_window(self.pop)
                self.upload_files()
            elif question is False:
                for _ in range(len(self.files)):
                    self.description.append("Κενό")
                self.upload_files()
        self.clear_buttons()
        self.clear_labels()
        self.change_values(self.controller)

    # Creation of loading screen when uploading,
    # creation of window for inserting a description to the files,
    # and then calling function files_for_upload() to start uploading
    def upload_files(self):
        try:
            self.progress_step = float(100.0 / len(self.files))
            anim = None
            # Creation of the loading screen
            def pop():
                global pop1, progressbar, files_label
                pop1 = tk.Toplevel(root)
                pop1.geometry('700x380+%s+%s' %
(int(pop1.winfo_screenwidth() / 2 - 700 / 2),
int(pop1.winfo_screenheight() / 2 - 380 / 2)))
                pop1.resizable(False, False)
                style = ttk.Style()
                style.configure('TFrame', background='white')
                frame = ttk.Frame(pop1, style='TFrame')
```

```
                frame.pack()
                pop1.focus_force()
                pop1.grab_set()
                title_label = ttk.Label(frame, text="Παρακαλώ
περιμένετε μέχρι να ολοκληρωθεί η διεργασία",
font=self.controller.title_font)
                title_label.grid(row=0, column=0, pady=30)
                file = "Spinner.gif"

                info = Image.open(file)

                frames = info.n_frames  # gives total number of
frames that gif contains

                # creating list of PhotoImage objects for each frames
                im = [tk.PhotoImage(file=file, format=f"gif -index
{ia}") for ia in range(frames)]

                count = 0
                style.configure('TLabel', background='white')
                gif_label = ttk.Label(frame, image="",
style='TLabel')
                gif_label.grid(row=1, column=0, pady=(0, 10))
                file_label = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="Αρχεία:\n")
                file_label.grid(row=1, column=0, pady=(0, 18))
                files_label = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="1/1")
                files_label.grid(row=1, column=0, pady=(8, 0))
                # make the gif move
                def animation(count):
                    global anim
                    im2 = im[count]

                    gif_label.configure(image=im2)
                    count += 1
                    if count == frames:
                        count = 0
                    anim = pop1.after(30, lambda: animation(count))

                animation(count)
                progressbar = ttk.Progressbar(frame,
mode='determinate', maximum=100, value=0)
                progressbar.grid(row=2, column=0, pady=(0, 30),
ipadx=100, ipady=3)

            def setv(value):
                progressbar["value"] += value

            # Constantly updates the window till bool becomes False
            def lp(bool):
                if bool:
                    pop1.update()

            def des():
                pop1.destroy()

            def run():
                count = 1
                for a in range(len(self.files)):
                    files_label.config(text="%d/%d" % (count,
```

```
len(self.files)))
                    self.files_for_upload(self.files[a],
self.description[a])
                    setv(self.progress_step)
                    count += 1

            pop()
            t1 = Thread(target=run, daemon=True)
            t1.start()
            while True:
                lp(True)
                if t1.is_alive() is False:
                    lp(False)
                    break
            des()
        except (FileNotFoundError, ZeroDivisionError, Exception) as
err:
            print("File error {}".format(err))
        finally:
            if len(self.get_recent_files()):
                self.clear_buttons()
                self.clear_labels()
                self.change_values(self.controller)
            else:
                self.clear_buttons()
                self.clear_labels()
                self.change_values(self.controller)

    # Calls insertBLOB() to upload the files chosen with their
informations
    def files_for_upload(self, file, description):
        filename = file.rsplit('/', 1)[-1]
        extension = filename.split('.', 1)[-1]
        date_time = "0-0-0 0:0:0"
        if extension.lower() in image_extensions:
            filetype_id = 2
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in text_extensions:
            filetype_id = 1
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in sound_extensions:
            filetype_id = 3
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in video_extensions:
            filetype_id = 4
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        else:
            filetype_id = 5
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
```

```
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        self.controller.frames["AdminPage"].update_time(date_time)
        self.description_upload(filename, description)

    # Uploads the file description to the database
    @staticmethod
    def description_upload(filename, description):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "INSERT INTO descriptions (files_id, description)
VALUES ((select max(id) from files where file_name = %s and user_id =
%s " \
                    "group by file_name), %s)"
            cursor.execute(query, (filename, int(row[0]),
description))
            cnx.commit()
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Creation of the window for inserting descriptions to files
    def description_pop(self, filename):
        self.pop = tk.Toplevel(root)

        def on_close():
            if messagebox.askyesno("Κενή περιγραφή", "Αν κλείσετε το
παράθυρο η περιγραφή θα είναι κενή. Είστε σίγουροι ότι θέλετε να
συνεχίσετε;"):
                self.description.append("Κενό")
                self.pop.destroy()

        self.pop.protocol("WM_DELETE_WINDOW", on_close)
        self.pop.geometry('600x320+%s+%s' %
(int(self.pop.winfo_screenwidth() / 2 - 600 / 2),
int(self.pop.winfo_screenheight() / 2 - 320 / 2)))
        self.pop.resizable(False, False)
        self.pop.iconbitmap('uowm-logo-big.ico')
        style = ttk.Style()
        style.configure('TFrame', background='white')
        frame = ttk.Frame(self.pop, style='TFrame')
        frame.pack()
        self.pop.focus_force()
        root.grab_release()
        self.pop.grab_set()
        label = tk.Label(frame, text="Εισάγωγη περιγραφής",
font=self.controller.title_font)
        label.grid(row=0, column=0)
        file_label = tk.Label(frame, text="Όνομα αρχείου: %s" %
```

```
filename, font=("Arial", 13))
        file_label.grid(row=1, column=0)
        text = scrolledtext.ScrolledText(frame, height=10, width=50,
spacing1=1, spacing3=1)
        text.grid(row=2, column=0)
        count_label = tk.Label(frame, text="Χαρακτήρες: 0/256")
        count_label.grid(row=3, column=0, padx=(0, 100), sticky="e")

        def count_characters(txt):
            t = txt.get(1.0, tk.END)
            if len(t) > 257:
                txt.delete('%s - 2c' % 'end')

        text.bind('<Any-KeyPress>', lambda e:
[count_characters(text), count_label.config(text="Χαρακτήρες: %s/256"
% (len(text.get(1.0, tk.END)) - 1))])
        text.bind('<Any-KeyRelease>',
                  lambda e: [count_characters(text),
count_label.config(text="Χαρακτήρες: %s/256" % (len(text.get(1.0,
tk.END)) - 1))],
                  add='+')

        def text_is_empty(txt):
            t = txt.get(1.0, '%s - 1c' % 'end')
            if t.strip() == '':
                self.description.append("Κενό")
            else:
                self.description.append(t)

        save = tk.Button(frame, text="Αποθήκευση", command=lambda:
[text_is_empty(text), self.pop.destroy()], height=1)
        save.grid(row=3, column=0, sticky="e")

    # Converts file from binary data to its normal format
    # in order to be downloaded
    @staticmethod
    def write_file(data, complete_path):
        # Convert binary data to proper format and write it on Hard
Disk
        try:
            with open(complete_path, 'wb') as file:
                file.write(data)
                messagebox.showinfo("Επιτυχής λήψη", "Η λήψη του
επιλεγμένου αρχείου ήταν επιτυχής.")
        except FileNotFoundError as err:
            if err.errno == 2:
                print("%s" % err)
            else:
                print("%s" % err)

    # Accessing database in order to download the specific file we
want
    def readBLOB(self, user_id, file_id):
        print("Reading BLOB data from files table")
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")
```

```
        cursor = cnx.cursor()

        try:
            sql_fetch_blob_query = """SELECT * FROM files WHERE
user_id = %s AND id = %s"""

            cursor.execute(sql_fetch_blob_query, (user_id, file_id))
            record = cursor.fetchall()
            for rows in record:
                file = rows[2]
                print("Storing file on disk \n")
                complete_path =
filedialog.asksaveasfilename(title="Διαλέξτε όνομα αρχείου",
filetypes=[("Όλα τα αρχεία", '*.*')],

initialfile=f"{str(rows[1])}")
                print(complete_path)
                self.write_file(file, complete_path)

        except mysql.connector.Error as error:
            print("Failed to read BLOB data from MySQL table
{}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Resize the HomePage to fit everything in the window
    def resize(self):
        self.grid(row=0, column=0, sticky="nsew")
        self.winfo_toplevel().geometry("")

    # Deletes the a file from database
    def delete_file(self, file_id, controller):
        print("Starting to delete record")
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            ask_delete = tk.messagebox.askyesno("Διαγραφή στοιχείου",
"Είστε σίγουροι ότι θέλετε να διαγράψετε αυτό το αρχείο;")
            if ask_delete is True:
                query = "DELETE FROM files WHERE files.id = %s"
                cursor.execute(query, (file_id,))
                cnx.commit()
                print("The record has been deleted")
                self.clear_labels()
                self.clear_buttons()
                self.change_values(controller)
            else:
                print("File not deleted")
        except mysql.connector.Error as err:
            print("MySQL Error: {}".format(err))
        finally:
            if cnx.is_connected():
```

```
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")


# Creation of frame that shows all the files the user has uploaded
# and the other users too
class ShowAllFiles(tk.Frame):
    # Initiates the frame ShowAllFiles
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        self.winfo_toplevel().geometry("")
        self.error_list = []
        search = StringVar()
        user_search = StringVar()
        self.page_count = StringVar()
        self.page_count.set("1")

        allTypes = self.get_all_file_types()

        self.currentType = StringVar()
        self.currentType.set("Όλα")
        self.currentType_user = StringVar()
        self.currentType_user.set("Όλα")

        a = ()
        a = list(a)
        for typ in allTypes:
            a.insert(typ[0], typ[1])
        a = tuple(a)

        self.fileTypesMenu = OptionMenu(self, self.currentType, *a,
command=self.type_selection_event)
        self.fileTypesMenu_user = OptionMenu(self,
self.currentType_user, *a, command=self.type_selection_user_event)
        description_font = tkfont.Font(size=8, family="Arial",
weight='bold')
        title_font = tkfont.Font(family='Helvetica', size=16,
weight="bold", slant="italic")
        title_label_user = tk.Label(self, text='Όλα τα αρχεία σας',
font=title_font)
        self.search_box_user = tk.Entry(self, width=45,
textvariable=user_search)
        title_label = tk.Label(self, text='Όλα τα αρχεία των άλλων
χρηστών', font=title_font)
        self.search_box = tk.Entry(self, width=45,
textvariable=search)

        def search_bind(button):
            if button.winfo_ismapped():
                button.invoke()

        self.search_box.bind('<Return>', lambda e:
search_bind(search_button1))
        self.search_box_user.bind('<Return>', lambda e:
search_bind(search_button))
        search_button = tk.Button(self, text="Αναζήτηση",
command=lambda: [self.user_clear_interior(),
self.user_search(user_search)])
        rename_button = tk.Button(self, text="Μετονομασία",
```

```
command=lambda:
self.user_change_file_name(self.user_check_box_identities_text))
        self.all_rename_button = tk.Button(self, text="Μετονομασία",
command=lambda:
self.all_change_file_name(self.check_box_identities_text))
        search_button1 = tk.Button(self, text="Αναζήτηση",
command=lambda: [self.all_clear_interior(),
self.all_user_search(search)])
        self.back_button = tk.Button(self, text='Πίσω',
command=lambda: [controller.frames["HomePage"].clear_buttons(),

controller.frames["HomePage"].clear_labels(),

controller.frames["HomePage"].change_values(controller),

controller.show_frame('HomePage'),

self.user_clear_interior(), self.all_clear_interior()])
        user_download_button = tk.Button(self, text='Λήψη',
command=lambda: self.submit(self.user_check_box_identities_text))
        user_delete_button = tk.Button(self, text='Διαγραφή',
command=lambda:
self.submit_delete(self.user_check_box_identities_text))
        user_upload_button = tk.Button(self, text='Ανέβασμα',
command=lambda: [self.upload(), self.check_error_list()])
        self.delete_button = tk.Button(self, text='Διαγραφή',
command=lambda:
self.all_submit_delete(self.check_box_identities_text))
        self.choose_all = tk.Button(self, text='Επιλογή όλων',
command=None)
        all_download_button = tk.Button(self, text='Λήψη',
command=lambda: self.submit(self.check_box_identities_text))
        self.choose_all1 = tk.Button(self, text='Επιλογή όλων',
command=None)
        description_frame = tk.LabelFrame(self, text="Περιγραφή",
bd=3, font=description_font)
        self.description =
tk.scrolledtext.ScrolledText(description_frame, height=10, width=50,
spacing1=1, spacing3=1, state=tk.DISABLED)
        self.edit_button = tk.Button(self, text="Ενημέρωση",
command=None)
        self.save_button = tk.Button(self, text='Αποθήκευση',
command=None)
        self.file_id = 0

        description_frame.grid(row=4, column=0, columnspan=3,
padx=10)
        self.description.grid(sticky='nswe', padx=2, pady=2)
        self.fileTypesMenu_user.grid(row=1, column=0, padx=(0, 280),
sticky='e')
        self.fileTypesMenu.grid(row=1, column=2, padx=(0, 280),
sticky='e')
        self.search_box_user.grid(row=1, column=0, sticky='e',
ipady=2.4)
        self.search_box.grid(row=1, column=2, sticky='e', ipady=2.4)
        search_button.grid(row=1, column=0, columnspan=1, sticky='e')
        rename_button.grid(row=1, column=0, sticky='w', padx=(13, 0))
        search_button1.grid(row=1, column=2, columnspan=1,
sticky='e')
        title_label_user.grid(row=0, column=0, columnspan=1)
        title_label.grid(row=0, column=2, columnspan=1)
```

```
        user_download_button.grid(row=3, column=0, sticky='e',
columnspan=1, ipadx=10, ipady=3)
        user_delete_button.grid(row=3, column=0, sticky='e',
columnspan=1, ipadx=10, ipady=3, padx=(0, 70))
        user_upload_button.grid(row=3, column=0, sticky='e',
columnspan=1, ipadx=10, ipady=3, padx=(0, 165))
        all_download_button.grid(row=3, column=2, sticky='e',
columnspan=1, ipadx=10, ipady=3)
        self.back_button.grid(row=5, column=2, columnspan=1,
sticky='e', ipadx=20, ipady=3)
        self.choose_all.grid(row=3, column=0, sticky='w', ipadx=10,
ipady=3)
        self.choose_all1.grid(row=3, column=2, sticky='w', ipadx=10,
ipady=3)
        self.edit_button.grid(row=5, column=0, ipady=3, columnspan=3,
padx=(350, 0))
        description_frame.grid_rowconfigure(0, weight=1)
        description_frame.grid_columnconfigure(0, weight=1)

        all_pages_label = tk.Label(self, text="Σελίδες")
        user_pages_label = tk.Label(self, text="Σελίδες")

        all_pages_label.grid(row=3, column=2, sticky="w", padx=(130,
0))
        user_pages_label.grid(row=3, column=0, sticky="w", padx=(130,
0))

        def activate():
            self.description.config(state=tk.NORMAL)
            self.description.focus_force()
            self.edit_button.grid_forget()
            self.save_button.grid(row=5, column=0, ipady=3,
columnspan=3, padx=(350, 0))

        def deactivate():
            self.description.config(state=tk.DISABLED)
            self.save_button.grid_forget()
            self.edit_button.grid(row=5, column=0, ipady=3,
columnspan=3, padx=(350, 0))

        self.save_button.config(command=lambda: [deactivate(),
self.save_description(self.file_id, self.description.get(1.0, '%s -
1c' % 'end'))])
        self.edit_button.config(command=lambda:  activate())

    def update_filename(self, file_name, file_id):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "UPDATE files SET file_name = %s WHERE id = %s"
            cursor.execute(query, (file_name, file_id))
            cnx.commit()
            print("File inserted successfully as a BLOB into files
table")
        except mysql.connector.Error as error:
```

```
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    def get_id(self, file_id):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "SELECT file_name FROM files WHERE id = %s"
            cursor.execute(query, (file_id, ))
            result = cursor.fetchone()
            print("File inserted successfully as a BLOB into files
table")
            return result
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    def user_change_file_name(self, checkbox_identities):
        result = [var.get() for var in checkbox_identities if
var.get()]
        if len(result):
            for res in result:
                f_name = self.get_id(res)
                if "." in str(f_name):
                    print("mphke sto if ")
                    global pop8
                    pop8 = tk.Toplevel(root)
                    pop8.title("Επεξεργασία ονόματος αρχείου")
                    pop8.iconbitmap('uowm-logo-big.ico')
                    change_name_label = tk.Label(pop8, text="Όνομα
αρχείου: %s" % f_name[0].rsplit(".", 1)[0], font=("Arial Black", 12))
                    if len(change_name_label["text"]) > 70:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:53] + "\n" +

f_name[0][53:].rsplit(".", 1)[0])
                    elif len(change_name_label["text"]) > 35:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:].rsplit(".", 1)[0])
                    change_name_label.pack(anchor="w", padx=(5, 0))
                    change_name_box = tk.Entry(pop8, width=50,
font=("Arial", 10), justify="left")
                    change_name_box.insert(0, f_name[0].rsplit(".",
1)[0])
                    change_name_box.pack(ipady=5)

                    def update_window():
```

```
                        self.user_clear_interior()

self.page_assigning_user(self.show_all_user())
                        self.user_page_fill(self.show_all_user())
                        self.file_id = 0
                        text = self.description
                        state = str(self.edit_button['state'])
                        if self.save_button.winfo_ismapped() == 1:
                            self.save_button.grid_forget()
                            self.edit_button.grid(row=5, column=0,
ipady=3, columnspan=3, padx=(350, 0))
                            text.config(state=tk.DISABLED)
                        if state == 'disabled':
                            self.edit_button.config(state=tk.NORMAL)
                            self.save_button.config(state=tk.NORMAL)
                        text.config(state=tk.NORMAL)
                        text.delete(1.0, tk.END)
                        text.config(state=tk.DISABLED)
                        self.currentType_user.set("Όλα")

                    update_button = tk.Button(pop8, text="Ενημέρωση",
                                              command=lambda:
[self.update_filename(change_name_box.get() + "." +
f_name[0].split('.', 1)[-1], res),

pop8.destroy()])
                    update_button.pack(side='left', padx=(120, 0),
ipady=2, pady=(5, 0))
                    cancel_button = tk.Button(pop8, text="Ακύρωση",
command=pop8.destroy)
                    cancel_button.pack(side='right', padx=(0, 120),
ipady=2, pady=(5, 0))
                    root.wait_window(pop8)
                    self.user_clear_interior()
                    self.page_assigning_user(self.show_all_user())
                    self.user_page_fill(self.show_all_user())
                    self.file_id = 0
                    text = self.description
                    state = str(self.edit_button['state'])
                    if self.save_button.winfo_ismapped() == 1:
                        self.save_button.grid_forget()
                        self.edit_button.grid(row=5, column=0,
ipady=3, columnspan=3, padx=(350, 0))
                        text.config(state=tk.DISABLED)
                    if state == 'disabled':
                        self.edit_button.config(state=tk.NORMAL)
                        self.save_button.config(state=tk.NORMAL)
                    text.config(state=tk.NORMAL)
                    text.delete(1.0, tk.END)
                    text.config(state=tk.DISABLED)
                else:
                    print("mphke sto else")
                    global pop10
                    pop10 = tk.Toplevel(root)
                    pop10.title("Επεξεργασία ονόματος αρχείου")
                    pop10.iconbitmap('uowm-logo-big.ico')
                    change_name_label = tk.Label(pop10, text="Όνομα
αρχείου: %s" % f_name[0], font=("Arial Black", 12))
                    if len(change_name_label["text"]) > 70:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:53] + "\n" +
```

```
f_name[0][53:])
                    elif len(change_name_label["text"]) > 35:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:])
                    change_name_label.pack(anchor="w", padx=(5, 0))
                    change_name_box = tk.Entry(pop10, width=50,
font=("Arial", 10), justify="left")
                    change_name_box.insert(0, f_name[0])
                    change_name_box.pack(ipady=5)

                    def update_window():
                        self.user_clear_interior()

self.page_assigning_user(self.show_all_user())
                        self.user_page_fill(self.show_all_user())
                        self.file_id = 0
                        text = self.description
                        state = str(self.edit_button['state'])
                        if self.save_button.winfo_ismapped() == 1:
                            self.save_button.grid_forget()
                            self.edit_button.grid(row=5, column=0,
ipady=3, columnspan=3, padx=(350, 0))
                            text.config(state=tk.DISABLED)
                        if state == 'disabled':
                            self.edit_button.config(state=tk.NORMAL)
                            self.save_button.config(state=tk.NORMAL)
                        text.config(state=tk.NORMAL)
                        text.delete(1.0, tk.END)
                        text.config(state=tk.DISABLED)
                        self.currentType_user.set("Όλα")

                    update_button = tk.Button(pop10,
text="Ενημέρωση",
                                              command=lambda:
[self.update_filename(change_name_box.get(), res), pop10.destroy(),
update_window()])
                    update_button.pack(side='left', padx=(120, 0),
ipady=2, pady=(5, 0))
                    cancel_button = tk.Button(pop10, text="Ακύρωση",
command=pop10.destroy)
                    cancel_button.pack(side='right', padx=(0, 120),
ipady=2, pady=(5, 0))
                    root.wait_window(pop10)
        else:
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για μετονομασία.")
        return

    def all_change_file_name(self, checkbox_identities):
        result = [var.get() for var in checkbox_identities if
var.get()]
        if len(result):
            for res in result:
                f_name = self.get_id(res)
                if "." in str(f_name):
                    global pop9
                    pop9 = tk.Toplevel(root)
                    pop9.title("Επεξεργασία ονόματος αρχείου")
                    pop9.iconbitmap('uowm-logo-big.ico')
                    change_name_label = tk.Label(pop9, text="Όνομα
αρχείου: %s" % f_name[0].rsplit(".", 1)[0], font=("Arial Black", 12))
```

```
                    if len(change_name_label["text"]) > 70:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:53] + "\n" +

f_name[0][53:].rsplit(".", 1)[0])
                    elif len(change_name_label["text"]) > 35:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:].rsplit(".", 1)[0])
                    change_name_label.pack(anchor="w", padx=(5, 0))
                    change_name_box = tk.Entry(pop9, width=50,
font=("Arial", 10), justify="left")
                    change_name_box.insert(0, f_name[0].rsplit(".",
1)[0])
                    change_name_box.pack(ipady=5)

                    def update_window():
                        self.all_clear_interior()
                        self.page_assigning(self.show_all())
                        self.page_fill(self.show_all())
                        self.file_id = 0
                        text = self.description
                        state = str(self.edit_button['state'])
                        if self.save_button.winfo_ismapped() == 1:
                            self.save_button.grid_forget()
                            self.edit_button.grid(row=5, column=0,
ipady=3, columnspan=3, padx=(350, 0))
                            text.config(state=tk.DISABLED)
                        if state == 'disabled':
                            self.edit_button.config(state=tk.NORMAL)
                            self.save_button.config(state=tk.NORMAL)
                        text.config(state=tk.NORMAL)
                        text.delete(1.0, tk.END)
                        text.config(state=tk.DISABLED)
                        self.currentType.set("Όλα")

                    update_button = tk.Button(pop9, text="Ενημέρωση",
                                              command=lambda:
[self.update_filename(change_name_box.get() + "." +
f_name[0].split('.', 1)[-1], res),

pop9.destroy(), update_window()])
                    update_button.pack(side='left', padx=(120, 0),
ipady=2, pady=(5, 0))
                    cancel_button = tk.Button(pop9, text="Ακύρωση",
command=pop9.destroy)
                    cancel_button.pack(side='right', padx=(0, 120),
ipady=2, pady=(5, 0))
                    root.wait_window(pop9)
                else:
                    global pop11
                    pop11 = tk.Toplevel(root)
                    pop11.title("Επεξεργασία ονόματος αρχείου")
                    pop11.iconbitmap('uowm-logo-big.ico')
                    change_name_label = tk.Label(pop11, text="Όνομα
αρχείου: %s" % f_name[0], font=("Arial Black", 12))
                    if len(change_name_label["text"]) > 70:
                        change_name_label.config(text="Όνομα αρχείου:
%s" % f_name[0][:20] + "\n" + f_name[0][20:53] + "\n" +
                                                 f_name[0][53:])
                    elif len(change_name_label["text"]) > 35:
                        change_name_label.config(text="Όνομα αρχείου:
```

```
%s" % f_name[0][:20] + "\n" + f_name[0][20:])
                    change_name_label.pack(anchor="w", padx=(5, 0))
                    change_name_box = tk.Entry(pop11, width=50,
font=("Arial", 10), justify="left")
                    change_name_box.insert(0, f_name[0])
                    change_name_box.pack(ipady=5)

                    def update_window():
                        self.all_clear_interior()
                        self.page_assigning(self.show_all())
                        self.page_fill(self.show_all())
                        self.file_id = 0
                        text = self.description
                        state = str(self.edit_button['state'])
                        if self.save_button.winfo_ismapped() == 1:
                            self.save_button.grid_forget()
                            self.edit_button.grid(row=5, column=0,
ipady=3, columnspan=3, padx=(350, 0))
                            text.config(state=tk.DISABLED)
                        if state == 'disabled':
                            self.edit_button.config(state=tk.NORMAL)
                            self.save_button.config(state=tk.NORMAL)
                        text.config(state=tk.NORMAL)
                        text.delete(1.0, tk.END)
                        text.config(state=tk.DISABLED)
                        self.currentType.set("Ολα")

                    update_button = tk.Button(pop11,
text="Ενημέρωση",
                                              command=lambda:
[self.update_filename(change_name_box.get(), res), pop11.destroy(),
update_window()])
                    update_button.pack(side='left', padx=(120, 0),
ipady=2, pady=(5, 0))
                    cancel_button = tk.Button(pop11, text="Ακύρωση",
command=pop11.destroy)
                    cancel_button.pack(side='right', padx=(0, 120),
ipady=2, pady=(5, 0))
                    root.wait_window(pop11)
        else:
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για μεονομασία.")
        return

    def check_error_list(self):
        if len(self.error_list):
            files = []
            for name in self.error_list:
                files.append(name.rsplit('/', 1)[-1])
            messagebox.showerror("Το αρχείο είναι πολύ μεγάλο",
                                 "Το αρχείο που προσπαθήσατε να
ανεβάσετε είναι μεγαλύτερο από το μέγιστο όριο."
                                 " \nΑρχείο: %s" % ', '.join(files))
        self.error_list = []

    # Converts the given file into binary data in order to
    # be uploaded in the Database
    def convertToBinaryData(self, filename):
        if os.path.getsize(filename) < 1048576:  # 16777216:
            with open(filename, 'rb') as file:
                binaryData = file.read()
```

```
            return binaryData
        else:
            self.error_list.append(filename)

        # Uploads the file to the Database

    def insertBLOB(self, filename, file, user_id, filetype_id,
date_added):
        print("Inserting BLOB into files table")
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            sql_insert_blob_query = "INSERT INTO files (file_name,
file, user_id, file_type_id, date_added) " \
                                    "VALUES (%s,%s,%s,%s,%s)"

            up_file = self.convertToBinaryData(file)

            # Convert data into tuple format
            insert_blob_tuple = (filename, up_file, user_id,
filetype_id, date_added)
            cursor.execute(sql_insert_blob_query, insert_blob_tuple)
            cnx.commit()
            print("File inserted successfully as a BLOB into files
table")
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Choose the files to be uploaded
    def upload(self):
        file_path = filedialog.askopenfilenames()
        self.files = list()
        self.description1 = list()
        for file in file_path:
            self.files.append(file)
        if len(self.files):
            question = messagebox.askyesnocancel("Περιγραφή
αρχείου/ων", "Πατήστε Yes, αν θέλετε να εισάγετε περιγραφή σε κάποιο
αρχείο\n"

"No, αν δεν θέλετε να εισαχθεί περιγραφή\nCancel, αν θέλετε να
ακυρώσετε το ανέβασμα"

" των αρχείων.")
            if question is True:
                for f in self.files:
                    self.description_pop(filename=f.rsplit('/', 1)[-
1])
                    root.wait_window(self.pop)
                self.upload_files()
```

```
            elif question is False:
                for _ in range(len(self.files)):
                    self.description1.append("Κενό")
                self.upload_files()

    # Creation of loading screen when uploading,
    # creation of window for inserting a description to the files,
    # and then calling function files_for_upload() to start uploading
    def upload_files(self):
        try:
            self.progress_step = float(100.0 / len(self.files))
            anim = None

            # Creation of the loading screen
            def pop():
                global pop7, progressbar7, files_label7
                pop7 = tk.Toplevel(root)
                pop7.geometry('700x380+%s+%s' %
(int(pop7.winfo_screenwidth() / 2 - 700 / 2),
int(pop7.winfo_screenheight() / 2 - 380 / 2)))
                pop7.resizable(False, False)
                style = ttk.Style()
                style.configure('TFrame', background='white')
                frame = ttk.Frame(pop7, style='TFrame')
                frame.pack()
                pop7.focus_force()
                pop7.grab_set()
                title_label = ttk.Label(frame, text="Παρακαλώ
περιμένετε μέχρι να ολοκληρωθεί η διεργασία",
font=self.controller.title_font)
                title_label.grid(row=0, column=0, pady=30)
                file = "Spinner.gif"

                info = Image.open(file)

                frames = info.n_frames  # gives total number of
frames that gif contains

                # creating list of PhotoImage objects for each frames
                im = [tk.PhotoImage(file=file, format=f"gif -index
{ia}") for ia in range(frames)]

                count = 0
                style.configure('TLabel', background='white')
                gif_label = ttk.Label(frame, image="",
style='TLabel')
                gif_label.grid(row=1, column=0, pady=(0, 10))
                file_label = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="Αρχεία:\n")
                file_label.grid(row=1, column=0, pady=(0, 18))
                files_label7 = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="1/1")
                files_label7.grid(row=1, column=0, pady=(8, 0))

                # make the gif move
                def animation(count):
                    global anim
                    im2 = im[count]

                    gif_label.configure(image=im2)
                    count += 1
```

```
                    if count == frames:
                        count = 0
                    anim = pop7.after(30, lambda: animation(count))

                animation(count)
                progressbar7 = ttk.Progressbar(frame,
mode='determinate', maximum=100, value=0)
                progressbar7.grid(row=2, column=0, pady=(0, 30),
ipadx=100, ipady=3)

            def setv(value):
                progressbar7["value"] += value

            # Constantly updates the window till bool becomes False
            def lp(bool):
                if bool:
                    pop7.update()

            def des():
                pop7.destroy()

            def run():
                count = 1
                for a in range(len(self.files)):
                    files_label7.config(text="%d/%d" % (count,
len(self.files)))
                    self.files_for_upload(self.files[a],
self.description1[a])
                    setv(self.progress_step)
                    count += 1

            pop()
            t1 = Thread(target=run, daemon=True)
            t1.start()
            while True:
                lp(True)
                if t1.is_alive() is False:
                    lp(False)
                    break
            des()
        except (FileNotFoundError, ZeroDivisionError, Exception) as
err:
            print("File error {}".format(err))
        finally:
            if len(self.show_all_user()):
                self.user_clear_interior()
                self.page_assigning_user(self.show_all_user())
                self.user_page_fill(self.show_all_user())
            else:
                self.user_clear_interior()
                self.page_assigning_user(self.show_all_user())
                self.user_page_fill(self.show_all_user())

        # Calls insertBLOB() to upload the files chosen with their
informations

    def files_for_upload(self, file, description):
        filename = file.rsplit('/', 1)[-1]
        extension = filename.split('.', 1)[-1]
        date_time = "0-0-0 0:0:0"
        if extension.lower() in image_extensions:
```

```
            filetype_id = 2
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in text_extensions:
            filetype_id = 1
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in sound_extensions:
            filetype_id = 3
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in video_extensions:
            filetype_id = 4
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        else:
            filetype_id = 5
            self.insertBLOB(filename, file, int(row[0]), filetype_id,
str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        self.controller.frames["AdminPage"].update_time(date_time)
        self.description_upload(filename, description)

        # Uploads the file description to the database

    @staticmethod
    def description_upload(filename, description):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "INSERT INTO descriptions (files_id, description)
VALUES ((select max(id) from files where file_name = %s and user_id =
%s " \
                    "group by file_name), %s)"
            cursor.execute(query, (filename, int(row[0]),
description))
            cnx.commit()
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Creation of the window for inserting descriptions to files
```

```
    def description_pop(self, filename):
        self.pop = tk.Toplevel(root)

        def on_close():
            if messagebox.askyesno("Κενή περιγραφή", "Αν κλείσετε το
παράθυρο η περιγραφή θα είναι κενή. Είστε σίγουροι ότι θέλετε να
συνεχίσετε;"):
                self.description1.append("Κενό")
                self.pop.destroy()

        self.pop.protocol("WM_DELETE_WINDOW", on_close)
        self.pop.geometry('600x320+%s+%s' %
(int(self.pop.winfo_screenwidth() / 2 - 600 / 2),
int(self.pop.winfo_screenheight() / 2 - 320 / 2)))
        self.pop.resizable(False, False)
        self.pop.iconbitmap('uowm-logo-big.ico')
        style = ttk.Style()
        style.configure('TFrame', background='white')
        frame = ttk.Frame(self.pop, style='TFrame')
        frame.pack()
        self.pop.focus_force()
        root.grab_release()
        self.pop.grab_set()
        label = tk.Label(frame, text="Εισάγωγη περιγραφής",
font=self.controller.title_font)
        label.grid(row=0, column=0)
        file_label = tk.Label(frame, text="Όνομα αρχείου: %s" %
filename, font=("Arial", 13))
        file_label.grid(row=1, column=0)
        text = scrolledtext.ScrolledText(frame, height=10, width=50,
spacing1=1, spacing3=1)
        text.grid(row=2, column=0)
        count_label = tk.Label(frame, text="Χαρακτήρες: 0/256")
        count_label.grid(row=3, column=0, padx=(0, 100), sticky="e")

        def count_characters(txt):
            t = txt.get(1.0, tk.END)
            if len(t) > 257:
                txt.delete('%s - 2c' % 'end')

        text.bind('<Any-KeyPress>',
                  lambda e: [count_characters(text),
count_label.config(text="Χαρακτήρες: %s/256" % (len(text.get(1.0,
tk.END)) - 1))])
        text.bind('<Any-KeyRelease>',
                  lambda e: [count_characters(text),
count_label.config(text="Χαρακτήρες: %s/256" % (len(text.get(1.0,
tk.END)) - 1))],
                  add='+')

        def text_is_empty(txt):
            t = txt.get(1.0, '%s - 1c' % 'end')
            if t.strip() == '':
                self.description1.append("Κενό")
            else:
                self.description1.append(t)

        save = tk.Button(frame, text="Αποθήκευση", command=lambda:
[text_is_empty(text), self.pop.destroy()], height=1)
        save.grid(row=3, column=0, sticky="e")
```

```
    # For every choice in the left dropdown menu calls function
type_search_user()
    def type_selection_user_event(self, event):
        all_types = self.get_all_file_types()
        if self.currentType_user.get() == all_types[1][1]:
            file_list_user = self.type_search_user(all_types[1][0])
        elif self.currentType_user.get() == all_types[0][1]:
            file_list_user = self.type_search_user(all_types[0][0])
        elif self.currentType_user.get() == all_types[2][1]:
            file_list_user = self.type_search_user(all_types[2][0])
        elif self.currentType_user.get() == all_types[3][1]:
            file_list_user = self.type_search_user(all_types[3][0])
        elif self.currentType_user.get() == all_types[4][1]:
            file_list_user = self.type_search_user(all_types[4][0])
        else:
            file_list_user = self.show_all_user()
        self.user_clear_interior()
        self.page_assigning_user(file_list_user)
        self.user_page_fill(file_list_user)

    # For every choice in the right dropdown menu calls function
type_search_user()
    def type_selection_event(self, event):
        all_types = self.get_all_file_types()
        if self.currentType.get() == all_types[1][1]:
            file_list = self.type_search_all(all_types[1][0])
        elif self.currentType.get() == all_types[0][1]:
            file_list = self.type_search_all(all_types[0][0])
        elif self.currentType.get() == all_types[2][1]:
            file_list = self.type_search_all(all_types[2][0])
        elif self.currentType.get() == all_types[3][1]:
            file_list = self.type_search_all(all_types[3][0])
        elif self.currentType.get() == all_types[4][1]:
            file_list = self.type_search_all(all_types[4][0])
        else:
            file_list = self.show_all()
        self.all_clear_interior()
        self.page_assigning(file_list)
        self.page_fill(file_list)

    # Defines how many pages of 50 checkboxes will be created for the
left side
    def page_assigning_user(self, file_list):
        self.user_pages = []
        count = len(file_list)
        pages_count = 1
        if count > 50:
            for i in range(count):
                if i % 50 == 0:
                    self.user_pages.append("%d" % pages_count)
                    pages_count += 1
        else:
            self.user_pages.append("%d" % pages_count)

    # Defines how many pages of 50 checkboxes will be created for the
right side
    def page_assigning(self, file_list):
        self.pages = []
        count = len(file_list)
        pages_count = 1
        if count > 50:
```

```
            for i in range(count):
                if i % 50 == 0:
                    self.pages.append("%d" % pages_count)
                    pages_count += 1
        else:
            self.pages.append("%d" % pages_count)

    # Calls function check_box_user() which creates the checkboxes
    # and creates the buttons for previous and next page for the left
side
    def user_page_fill(self, file_list):
        self.check_box_user(file_list, 1)
        prev_button = tk.Button(self, text='<', command=None,
state=tk.DISABLED)
        next_button = tk.Button(self, text='>', command=None)
        page_label = tk.Label(self, text='%d/%d' %
(int(self.user_pages[0]), len(self.user_pages)), width=4)

        page_label.grid(row=3, column=0, sticky='w', padx=(194, 0))
        prev_button.grid(row=3, column=0, sticky='w', padx=(174, 0))
        next_button.grid(row=3, column=0, sticky='w', padx=(228, 0))

        def prev_page(page):
            next_button.config(state=tk.NORMAL, command=lambda:
[self.user_clear_interior(), next_page(page + 1)])
            prev_button.config(state=tk.NORMAL, command=lambda:
[self.user_clear_interior(), prev_page(page - 1)])

            if page < 0:
                prev_button.config(state=tk.DISABLED)
            page_label.config(text='%d/%d' % (abs(page + 2),
len(self.user_pages)))
            Thread(self.check_box_user(file_list, page + 2)).start()
            self.choose_all.config(text="Επιλογή όλων")

        def next_page(page):
            next_button.config(state=tk.NORMAL, command=lambda:
[self.user_clear_interior(), next_page(page + 1)])
            prev_button.config(state=tk.NORMAL, command=lambda:
[self.user_clear_interior(), prev_page(page - 1)])

            if page == len(self.user_pages) - 2:
                next_button.config(state=tk.DISABLED)
            page_label.config(text='%d/%d' % (abs(page + 2),
len(self.user_pages)))
            Thread(self.check_box_user(file_list, page + 2)).start()
            self.choose_all.config(text="Επιλογή όλων")

        prev_button.config(command=prev_page)
        if len(self.user_pages) == 1:
            next_button.config(state=tk.DISABLED)
        next_button.config(command=lambda:
[self.user_clear_interior(), next_page(0)])

    # Calls function check_box_user() which creates the checkboxes
    # and creates the buttons for previous and next page for the
rigth side
    def page_fill(self, file_list):
        self.all_check_box(file_list, 1)
        prev_button = tk.Button(self, text='<', command=None,
state=tk.DISABLED)
```

```
        next_button = tk.Button(self, text='>', command=None)
        page_label = tk.Label(self, text='%d/%d' %
(int(self.pages[0]), len(self.pages)), width=4)

        page_label.grid(row=3, column=2, sticky='w', padx=(194, 0))
        prev_button.grid(row=3, column=2, sticky='w', padx=(174, 0))
        next_button.grid(row=3, column=2, sticky='w', padx=(228, 0))

        def prev_page(page):
            next_button.config(state=tk.NORMAL, command=lambda:
[self.all_clear_interior(), next_page(page + 1)])
            prev_button.config(state=tk.NORMAL, command=lambda:
[self.all_clear_interior(), prev_page(page - 1)])

            if page < 0:
                prev_button.config(state=tk.DISABLED)
            page_label.config(text='%d/%d' % (abs(page + 2),
len(self.pages)))
            Thread(self.all_check_box(file_list, page + 2)).start()
            self.choose_all1.config(text="Επιλογή όλων")

        def next_page(page):
            next_button.config(state=tk.NORMAL, command=lambda:
[self.all_clear_interior(), next_page(page + 1)])
            prev_button.config(state=tk.NORMAL, command=lambda:
[self.all_clear_interior(), prev_page(page - 1)])

            if page == len(self.pages) - 2:
                next_button.config(state=tk.DISABLED)
            page_label.config(text='%d/%d' % (abs(page + 2),
len(self.pages)))
            Thread(self.all_check_box(file_list, page + 2)).start()
            self.choose_all1.config(text="Επιλογή όλων")

        prev_button.config(command=prev_page)
        if len(self.pages) == 1:
            next_button.config(state=tk.DISABLED)
        next_button.config(command=lambda:
[self.all_clear_interior(), next_page(0)])

    # Returns all user's files
    @staticmethod
    def show_all_user():
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()
            show_all_query = "SELECT files.file_name,
files.date_added, files.id FROM files,users WHERE users.id = %s AND
files.user_id = %s" \
                             " ORDER BY files.date_added DESC,
files.id DESC"
            cursor.execute(show_all_query, (int(row[0]),
(int(row[0]))))
            result = cursor.fetchall()
            return result
```

```
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected() is True:
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Returns all users' files
    @staticmethod
    def show_all():
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()
            show_all_query = "SELECT files.file_name ,
files.date_added, users.name, files.id FROM files " \
                             "INNER JOIN users ON users.id =
files.user_id AND files.user_id <> %s " \
                             "ORDER BY files.date_added DESC,
files.id DESC"
            cursor.execute(show_all_query, (int(row[0]),))
            result = cursor.fetchall()
            return result
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected() is True:
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Creates the checkboxes in the left side
    def check_box_user(self, file_list, page):
        # create a canvas object and a vertical scrollbar for
scrolling it
        vscrollbar = tk.Scrollbar(self, orient='vertical')
        vscrollbar.grid(row=2, column=1, ipady=123, padx=(0, 5),
sticky='nsew')
        canvas = tk.Canvas(self, bd=1, highlightthickness=0,
                           yscrollcommand=vscrollbar.set)
        canvas.grid(row=2, column=0, ipadx=10, ipady=16, padx=(10,
0), sticky='nsew')
        vscrollbar.config(command=canvas.yview)
        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled
with it
        self.user_interior = user_interior = tk.Listbox(canvas)
        interior_id = canvas.create_window(0, 0,
window=user_interior,
                                           anchor='nw')

        # track changes to the canvas and frame width and sync them,
```

```
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner
frame
            size = (user_interior.winfo_reqwidth(),
user_interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=user_interior.winfo_reqwidth())

        user_interior.bind('<Configure>', _configure_interior)

        self.user_label_identities = []
        file_name_label = tk.Label(self.user_interior, text="Ονομα
αρχείου", width=50)
        file_name_label.grid(row=0, column=0, sticky='w', padx=(0,
30))
        date_label = tk.Label(self.user_interior, text="Ημερομηνία")
        date_label.grid(row=0, column=1, sticky='nsew')
        self.user_label_identities.append((file_name_label,
date_label))

        self.user_check_box_identities = []
        self.user_check_box_identities_text = []
        self.date_identities = []
        counter = 1
        count = 1
        num1 = 50 * (page - 1)
        num2 = 50 * page
        if page == len(self.user_pages):
            for rows in range(num1, len(file_list)):
                label_text = file_list[rows]
                var = StringVar(value=label_text[2])
                if len(label_text[0]) > 50:
                    check_box = tk.Checkbutton(self.user_interior,
variable=var, onvalue=label_text[2], offvalue='',

text=label_text[0][:50] + "\n" + label_text[0][50:],
command=partial(self.user_description_fill, var))
                else:
                    check_box = tk.Checkbutton(self.user_interior,
variable=var, onvalue=label_text[2], offvalue='', text=label_text[0],

command=partial(self.user_description_fill, var))
                check_box.deselect()
                self.user_check_box_identities_text.append(var)
                self.user_check_box_identities.append(check_box)
                label_date = tk.Label(self.user_interior,
text=label_text[1])
                self.date_identities.append(label_date)

                label_date.grid(row=counter, column=1)
                check_box.grid(row=counter, column=0, sticky='w')
                counter += 1
                count += 1
        else:
            for rows in range(num1, num2):
                label_text = file_list[rows]
                var = StringVar(value=label_text[2])
```

```
                if len(label_text[0]) > 50:
                    check_box = tk.Checkbutton(self.user_interior,
variable=var, onvalue=label_text[2], offvalue='',

text=label_text[0][:50] + "\n" + label_text[0][50:],
command=partial(self.user_description_fill, var))
                else:
                    check_box = tk.Checkbutton(self.user_interior,
variable=var, onvalue=label_text[2], offvalue='', text=label_text[0],

command=partial(self.user_description_fill, var))
                check_box.deselect()
                self.user_check_box_identities_text.append(var)
                self.user_check_box_identities.append(check_box)
                label_date = tk.Label(self.user_interior,
text=label_text[1])
                self.date_identities.append(label_date)

                label_date.grid(row=counter, column=1)
                check_box.grid(row=counter, column=0, sticky='w')
                counter += 1
                count += 1

        def _configure_canvas(event):
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id,
width=canvas.winfo_width())

        canvas.bind('<Configure>', _configure_canvas)

        def select_all():
            self.choose_all.config(text="Αποεπιλογή όλων",
command=deselect_all)
            for i in self.user_check_box_identities:
                i.select()

        def deselect_all():
            self.choose_all.config(text="Επιλογή όλων",
command=select_all)
            for i in self.user_check_box_identities:
                i.deselect()
            if len(self.description.get(1.0, tk.END)):
                self.description.config(state=tk.NORMAL)
                self.description.delete(1.0, tk.END)
                self.description.config(state=tk.DISABLED)

        self.choose_all.config(command=select_all)

    # Creates the checkboxes in the right side
    def all_check_box(self, file_list, page):
        # create a canvas object and a vertical scrollbar for
scrolling it
        vscrollbar = tk.Scrollbar(self, orient='vertical')
        vscrollbar.grid(row=2, column=3, ipady=123, padx=(0, 5),
sticky='nsew')
        canvas = tk.Canvas(self, bd=1, highlightthickness=0,
                           yscrollcommand=vscrollbar.set)
        canvas.grid(row=2, column=2, ipadx=10, ipady=16, padx=(10,
0), sticky='nsew')
```

```
        vscrollbar.config(command=canvas.yview)
        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled
with it
        self.interior = interior = tk.Listbox(canvas)
        interior_id = canvas.create_window(0, 0, window=interior,
                                           anchor='nw')

        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner
frame
            size = (interior.winfo_reqwidth(),
interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=interior.winfo_reqwidth())

        interior.bind('<Configure>', _configure_interior)

        self.all_label_identities = []
        file_name_label = tk.Label(self.interior, text="Όνομα
αρχείου", width=50)
        username_label = tk.Label(self.interior, text="Χρήστης")
        date_label = tk.Label(self.interior, text="Ημερομηνία")
        self.all_label_identities.append((file_name_label,
username_label, date_label))

        file_name_label.grid(row=0, column=0, sticky='w')
        username_label.grid(row=0, column=1, sticky='nsew', ipadx=60)
        date_label.grid(row=0, column=2, sticky='nsew')

        self.check_box_identities = []
        self.check_box_identities_text = []
        self.name_date_identities = []

        counter = 1
        count = 1
        num1 = 50 * (page - 1)
        num2 = 50 * page
        if page == len(self.pages):
            for rows in range(num1, len(file_list)):
                label_text = file_list[rows]
                var = StringVar()
                if len(label_text[0]) <= 50:
                    check_box = tk.Checkbutton(self.interior,
variable=var, onvalue=label_text[3], offvalue='', text=label_text[0],

command=partial(self.description_fill, var))
                else:
                    check_box = tk.Checkbutton(self.interior,
variable=var, onvalue=label_text[3], offvalue='',

text=label_text[0][:50] + "\n" + label_text[0][50:],
command=partial(self.description_fill, var))
                check_box.deselect()
```

```
                self.check_box_identities_text.append(var)
                self.check_box_identities.append(check_box)
                label = tk.Label(self.interior, text=label_text[2])
                date_time_label = tk.Label(self.interior,
text=label_text[1])
                self.name_date_identities.append((label,
date_time_label))

                label.grid(row=counter, column=1)
                date_time_label.grid(row=counter, column=2)
                check_box.grid(row=counter, column=0, sticky='w')
                counter += 1
                count += 1
        else:
            for rows in range(num1, num2):
                label_text = file_list[rows]
                var = StringVar()
                if len(label_text[0]) <= 50:
                    check_box = tk.Checkbutton(self.interior,
variable=var, onvalue=label_text[3], offvalue='', text=label_text[0],

command=partial(self.description_fill, var))
                else:
                    check_box = tk.Checkbutton(self.interior,
variable=var, onvalue=label_text[3], offvalue='',

text=label_text[0][:50] + "\n" + label_text[0][50:],
command=partial(self.description_fill, var))
                check_box.deselect()
                self.check_box_identities_text.append(var)
                self.check_box_identities.append(check_box)
                label = tk.Label(self.interior, text=label_text[2])
                date_time_label = tk.Label(self.interior,
text=label_text[1])
                self.name_date_identities.append((label,
date_time_label))

                label.grid(row=counter, column=1)
                date_time_label.grid(row=counter, column=2)
                check_box.grid(row=counter, column=0, sticky='w')
                counter += 1
                count += 1

        def _configure_canvas(event):
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id,
width=canvas.winfo_width())

        canvas.bind('<Configure>', _configure_canvas)

        def select_all():
            self.choose_all1.config(text="Αποεπιλογή όλων",
command=deselect_all)
            for i in self.check_box_identities:
                i.select()

        def deselect_all():
            self.choose_all1.config(text="Επιλογή όλων",
command=select_all)
            for i in self.check_box_identities:
```

```
                i.deselect()
            if len(self.description.get(1.0, tk.END)):
                self.description.config(state=tk.NORMAL)
                self.description.delete(1.0, tk.END)
                self.description.config(state=tk.DISABLED)

        self.choose_all1.config(command=select_all)

    # Uploads the description in Database
    def save_description(self, id, description):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "UPDATE descriptions SET description = %s WHERE
files_id = %s"
            cursor.execute(query, (description, id))
            cnx.commit()
        except mysql.connector.Error as err:
            print("MySQL connection error: {}".format(err))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()

    # Fills the description textbox with the file description
    # for the files of the user
    def user_description_fill(self, var):
        text = self.description
        state = str(self.edit_button['state'])
        if self.save_button.winfo_ismapped() == 1:
            self.save_button.grid_forget()
            self.edit_button.grid(row=5, column=0, ipady=3,
columnspan=3, padx=(350, 0))
            text.config(state=tk.DISABLED)
        if state == 'disabled':
            self.edit_button.config(state=tk.NORMAL)
            self.save_button.config(state=tk.NORMAL)
        if var.get().strip() != '':
            self.file_id = int(var.get())
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()
            text.config(state=tk.NORMAL)
            text.delete(1.0, tk.END)
            text.config(state=tk.DISABLED)
            try:
                query = "SELECT description FROM descriptions WHERE
files_id = %s"
                cursor.execute(query, (var.get(),))
                result = cursor.fetchone()
```

```
                text.config(state=tk.NORMAL)
                text.insert(tk.INSERT, result[0])
                text.config(state=tk.DISABLED)
            except mysql.connector.Error as error:
                print("MySQL Error: {}".format(error))
            finally:
                if cnx.is_connected() is True:
                    cursor.close()
                    cnx.close()
                    print("MySQL connection is closed")
                    text.config(state=tk.DISABLED)
        else:
            text.config(state=tk.NORMAL)
            text.delete(1.0, tk.END)
            text.config(state=tk.DISABLED)
            self.file_id = None

    # Fills the description textbox with the file description
    # for the files of all users
    def description_fill(self, var):
        text = self.description
        state = str(self.edit_button['state'])
        if self.save_button.winfo_ismapped() == 1:
            self.save_button.grid_forget()
            self.edit_button.grid(row=5, column=0, ipady=3,
columnspan=3, padx=(350, 0))
            text.config(state=tk.DISABLED)
        if row[1] == 0:
            if state != 'disabled':
                self.edit_button.config(state=tk.DISABLED)
                self.save_button.config(state=tk.DISABLED)

        if var.get().strip() != '':
            self.file_id = int(var.get())
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()
            text.config(state=tk.NORMAL)
            text.delete(1.0, tk.END)
            text.config(state=tk.DISABLED)
            try:
                query = "SELECT description FROM descriptions WHERE
files_id = %s"
                cursor.execute(query, (var.get(),))
                result = cursor.fetchone()
                text.config(state=tk.NORMAL)
                text.insert(tk.INSERT, result[0])
                text.config(state=tk.DISABLED)
            except (mysql.connector.Error, TypeError) as error:
                if error.__class__.__name__ ==
"mysql.connector.Error":
                    print("MySQL Error: {}".format(error))
                else:
                    print("Error: {}".format(error))
            finally:
                if cnx.is_connected() is True:
```

```
                    cursor.close()
                    cnx.close()
                    print("MySQL connection is closed")
                    text.config(state=tk.DISABLED)
        else:
            text.config(state=tk.NORMAL)
            text.delete(1.0, tk.END)
            text.config(state=tk.DISABLED)
            self.file_id = None

    # Searches for what's written ih the left search field
    def user_search(self, user_search):
        file_list = self.show_all_user()
        user_search = user_search.get()
        if user_search == '' or str.isspace(user_search) is True:
            self.page_assigning_user(file_list)
            self.user_page_fill(file_list)
        else:
            user_files = []
            for file in file_list:
                if user_search in file[0]:
                    user_files.append(file)
                else:
                    continue
            self.page_assigning_user(user_files)
            self.user_page_fill(user_files)

    # Destroys everything in the right canvas
    def all_clear_interior(self):
        if len(self.check_box_identities):
            for c in self.check_box_identities:
                c.destroy()
        if len(self.all_label_identities):
            for l1, l2, l3 in self.all_label_identities:
                l1.destroy()
                l2.destroy()
                l3.destroy()
        if len(self.name_date_identities):
            for n, d in self.name_date_identities:
                n.destroy()
                d.destroy()
        self.interior.destroy()

    # Destroys everything in the left canvas
    def user_clear_interior(self):
        if len(self.user_check_box_identities):
            for c in self.user_check_box_identities:
                c.destroy()
        if len(self.user_label_identities):
            for l1, l2 in self.user_label_identities:
                l1.destroy()
                l2.destroy()
        if len(self.date_identities):
            for d in self.date_identities:
                d.destroy()
        self.user_interior.destroy()

    # Searches for what's written ih the right search field
    def all_user_search(self, search):
        search = search.get()
        file_list = self.show_all()
```

```
        if search == '' or str.isspace(search) is True:
            self.page_assigning(file_list)
            self.page_fill(file_list)
        else:
            files = []
            for file in self.show_all():
                if search in file[0]:
                    files.append(file)
                else:
                    continue
            self.page_assigning(files)
            self.page_fill(files)

    # Convert binary data to proper format and write it on Hard Disk
    @staticmethod
    def write_file(data, complete_path):
        with open(complete_path, 'wb') as file:
            file.write(data)

    # Accessing database in order to download the specific file we
want
    def readBLOB(self, file_id):
        print("Reading BLOB data from files table")

        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()
            sql_fetch_blob_query = """SELECT file_name, file FROM
files WHERE id = %s"""

            cursor.execute(sql_fetch_blob_query, (file_id,))
            record = cursor.fetchall()
            return record
        except mysql.connector.Error as error:
            print("Failed to read BLOB data from MySQL table
{}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    def single_download_file(self, record, complete_path):
        for rows in record:
            file = rows[1]
            self.write_file(file, complete_path)
            messagebox.showinfo("Επιτυχής λήψη", "Η λήψη του
επιλεγμένου αρχείου ήταν επιτυχής.")

    def multiple_download_file(self, record, complete_path):
        for rows in record:
            file = rows[1]
            if os.path.isfile(complete_path + '/' + str(rows[0])) is
False:
                self.write_file(file, complete_path + '/' +
```

```
str(rows[0]))
                return True
            else:
                question = messagebox.askyesno("Το αρχείο υπάρχει
ήδη", "Το αρχείο που κατέβαζετε υπάρχει ήδη στον φάκελο. "

"Αν επιθυμείτε να το μετονομάσετε πατήστε Yes, αλλιώς αν θέλετε να "

"διαγραφεί το υπάρχον αρχείο και να διατηρήσετε το καινούργιο πατήστε
No.")
                if question is True:
                    new_complete_path =
filedialog.asksaveasfilename(title="Διαλέξτε όνομα αρχείου",
filetypes=[("Όλα τα αρχεία", '*.*')],

initialfile=f"{str(rows[0])}")
                    self.write_file(file, new_complete_path)
                    return True
                else:
                    self.write_file(file, complete_path + '/' +
str(rows[0]))
                    return True

    # Function for the download buttons on both sides
    def submit(self, check_box_identities_text):
        # extract roll numbers for checked checkbuttons
        result = [var.get() for var in check_box_identities_text if
var.get()]
        if len(result) == 1:
            path = filedialog.asksaveasfilename(title="Διαλέξτε όνομα
αρχείου", filetypes=[("Όλα τα αρχεία", '*.*')],

initialfile=f"{self.readBLOB(result[0])[0][0]}")
            if len(result) == 1 and path != '':
                self.single_download_file(self.readBLOB(result[0]),
path)
        elif len(result) > 1:
            path = filedialog.askdirectory()
            if len(result) and path != '':
                self.progress_step = float(100.0 / len(result))
                anim = None

                # Creation of the loading screen
                def pop():
                    global pop6, progressbar6, files_label6
                    pop6 = tk.Toplevel(root)
                    pop6.geometry(
                        '700x380+%s+%s' %
(int(pop6.winfo_screenwidth() / 2 - 700 / 2),
int(pop6.winfo_screenheight() / 2 - 380 / 2)))
                    pop6.resizable(False, False)
                    style = ttk.Style()
                    style.configure('TFrame', background='white')
                    frame = ttk.Frame(pop6, style='TFrame')
                    frame.pack()
                    pop6.focus_force()
                    pop6.grab_set()
                    title_label = ttk.Label(frame, text="Παρακαλώ
περιμένετε μέχρι να ολοκληρωθεί η διεργασία",

font=self.controller.title_font)
```

```
                    title_label.grid(row=0, column=0, pady=30)
                    file = "Spinner.gif"

                    info = Image.open(file)

                    frames = info.n_frames  # gives total number of
frames that gif contains

                    # creating list of PhotoImage objects for each
frames
                    im = [tk.PhotoImage(file=file, format=f"gif -
index {ia}") for ia in range(frames)]

                    count = 0
                    style.configure('TLabel', background='white')
                    gif_label = ttk.Label(frame, image="",
style='TLabel')
                    gif_label.grid(row=1, column=0, pady=(0, 10))
                    file_label = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="Αρχεία:\n")
                    file_label.grid(row=1, column=0, pady=(0, 18))
                    files_label6 = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="1/1")
                    files_label6.grid(row=1, column=0, pady=(8, 0))

                    # make the gif move
                    def animation(count):
                        global anim
                        im2 = im[count]

                        gif_label.configure(image=im2)
                        count += 1
                        if count == frames:
                            count = 0
                        anim = pop6.after(30, lambda:
animation(count))

                    animation(count)
                    progressbar6 = ttk.Progressbar(frame,
mode='determinate', maximum=100, value=0)
                    progressbar6.grid(row=2, column=0, pady=(0, 30),
ipadx=100, ipady=3)

                def setv(value):
                    progressbar6["value"] += value

                # Constantly updates the window till bool becomes
False
                def lp(bool):
                    if bool:
                        pop6.update()

                def des():
                    pop6.destroy()

                def run():
                    count = 1
                    for a in range(len(result)):
                        files_label6.config(text="%d/%d" % (count,
len(result)))
```

```
self.multiple_download_file(self.readBLOB(result[a]), path)
                        setv(self.progress_step)
                        count += 1
                pop()
                t1 = Thread(target=run, daemon=True)
                t1.start()
                while True:
                    lp(True)
                    if t1.is_alive() is False:
                        lp(False)
                        break
                des()
            else:
                print("Empty path or no files were selected")
        else:
            print("No files were selected")
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για λήψη.")

    # Deletes a file from database
    @staticmethod
    def delete_file(file_id):
        print("Starting to delete record")
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()

            query = "DELETE FROM files WHERE files.id = %s"
            cursor.execute(query, (file_id,))
            cnx.commit()
            print("The record has been deleted")
        except mysql.connector.Error as err:
            print("MySQL Error: {}".format(err))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Function for the left delete button
    def submit_delete(self, checkbox_identities):
        result = [var.get() for var in checkbox_identities if
var.get()]
        if len(result):
            ask_delete = tk.messagebox.askyesno("Διαγραφή αρχείων",
"Είστε σίγουροι ότι θέλετε να διαγράψετε τα επιλεγμένα αρχεία;")
            if ask_delete is True:
                with concurrent.futures.ThreadPoolExecutor() as
executor:
                    executor.map(self.delete_file, result)
                file_list = self.show_all_user()
                self.choose_all.config(text="Επιλογή όλων")
                self.user_clear_interior()
                self.page_assigning_user(file_list)
                self.user_page_fill(file_list)
```

```
                self.description.config(state=tk.NORMAL)
                self.description.delete(1.0, tk.END)
                self.description.config(state=tk.DISABLED)
            else:
                print("File not deleted")
        else:
            print("no files were selected")
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για διαγραφή.")

    #Function for the right delete button
    def all_submit_delete(self, checkbox_identities):
        result = [var.get() for var in checkbox_identities if
var.get()]
        if len(result):
            ask_delete = tk.messagebox.askyesno("Διαγραφή αρχείων",
"Είστε σίγουροι ότι θέλετε να διαγράψετε τα επιλεγμένα αρχεία;")
            if ask_delete is True:
                with concurrent.futures.ThreadPoolExecutor() as
executor:
                    executor.map(self.delete_file, result)
                file_list = self.show_all()
                self.choose_all.config(text="Επιλογή όλων")
                self.all_clear_interior()
                self.page_assigning(file_list)
                self.page_fill(file_list)
                self.description.config(state=tk.NORMAL)
                self.description.delete(1.0, tk.END)
                self.description.config(state=tk.DISABLED)
            else:
                print("File not deleted")
        else:
            print("no files were selected")
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για διαγραφή.")

    # Function that returns all the filetypes in the database
    @staticmethod
    def get_all_file_types():
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cur = cnx.cursor()
            cur.execute("SELECT id,name FROM file_types")
            rows = cur.fetchall()
            return rows
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Returns all user files by the type
    # chosen from the left dropdown menu
```

```
    @staticmethod
    def type_search_user(type_id):
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cur = cnx.cursor()
            cur.execute("SELECT files.file_name, files.date_added,
files.id FROM files,users "
                        f"WHERE files.user_id = {int(row[0])} AND
users.id = {int(row[0])} AND files.file_type_id = {int(type_id)} "
                        "ORDER BY files.date_added DESC, files.id
DESC")
            rows = cur.fetchall()
            return rows
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Returns all users files by the type
    # chosen from the right dropdown menu
    @staticmethod
    def type_search_all(type_id):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cur = cnx.cursor()
        try:
            cur.execute("SELECT files.file_name , files.date_added,
users.name, files.id FROM files,users "
                        f"WHERE files.user_id <> {int(row[0])} AND
files.file_type_id = {int(type_id)} AND users.id = files.user_id "
                        "ORDER BY files.date_added DESC, files.id
DESC")
            rows = cur.fetchall()
            return rows
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")


# The starting window of the app
class AuthenticationForm(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
```

```
        self.controller = controller

        label = tk.Label(self, text="Συνδεθείτε με το λογαριασμό
σας", relief='flat', font=controller.title_font)
        label.pack(side="top", fill="x", pady=15)

        button1 = tk.Button(self, text="Σύνδεση",
                            command=lambda:
controller.show_frame("LoginForm"), width="30", height="2")

        tk.Label(text="", pady="10").pack()

        button2 = tk.Button(self, text="Εγγραφή",
                            command=lambda:
controller.show_frame("RegisterForm"), width="30", height="2")
        button1.pack(pady=30)
        button2.pack(pady=10)


# The window of login
class LoginForm(tk.Frame):
    # Method that checks the login credentials
    @staticmethod
    def login(controller, login_username, login_password):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")
        cur = cnx.cursor()
        try:
            global row
            login_username = login_username.get()
            login_password = login_password.get()

            query = f"SELECT id, access_level FROM users WHERE
password=%s AND email=%s"
            cur.execute(query, (login_password, login_username))
            row = cur.fetchone()
            if str.strip(login_username) == "" or
str.strip(login_password) == "":
                messagebox.showerror("Υπάρχουν κενά πεδια",
                                     "Παρακαλώ συμπληρώστε όλα τα
κενά πεδία.")
                return
            elif row is None:
                cnx.close()
                messagebox.showerror("Λάθος στοιχεία",
                                     "Τα στοιχεία που συμπληρώσατε
δεν αντιστοιχούν με κανένα λογαριασμό.")
                return
            if int(row[1]) == 0:
                print(row[1])

controller.frames["HomePage"].change_values(controller)
                controller.show_frame("HomePage")
            else:
                print(row[1])
                controller.show_frame("AdminPage")
                admin_width = int(root.winfo_screenwidth() / 2 - 840
```

```
/ 2)
                admin_height = int(root.winfo_screenheight() / 2 -
500 / 2)
                root.geometry("840x405+%s+%s" % (admin_width,
admin_height))

controller.frames["AdminPage"].check_box_users(controller.frames["Adm
inPage"].show_users())
                controller.frames["AdminPage"].recent_files_chk_box()
        except mysql.connector.Error as err:
            print("Error connecting to MySQL {}".format(err))
        finally:
            if cnx.is_connected() is True:
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Constructor of this class
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        global login_entry
        global password_entry
        label = tk.Label(self, text="Παρακαλώ συνδεθείτε με τα
στοιχεία σας", font=controller.title_font)
        label.grid(row=0, column=0, sticky="new", pady=10)

        button = tk.Button(self, text="Πίσω")

        labelEmail = tk.Label(self, text="Email")
        labelEmail.grid(row=1, column=0)

        login_username = StringVar()
        login_password = StringVar()

        login_entry = tk.Entry(self, width=40,
textvariable=login_username, justify='center')
        login_entry.grid(row=2, column=0, pady=(0, 10))

        labelPassword = tk.Label(self, text="Κωδικός Πρόσβασης")
        labelPassword.grid(row=3, column=0)

        password_entry = tk.Entry(self, width=30,
textvariable=login_password, show='●', justify='center')
        password_entry.grid(row=4, column=0)
        photo = Image.open("show-password-icon-18.jpg")
        photo1 = Image.open("show-password-icon-17.jpg")
        resized_photo = photo.resize((20, 20), Image.ANTIALIAS)
        resized_photo1 = photo1.resize((20, 20), Image.ANTIALIAS)
        show_photo = ImageTk.PhotoImage(resized_photo)
        show_photo1 = ImageTk.PhotoImage(resized_photo1)
        show_button = tk.Button(self, image=show_photo, width=15,
height=15, relief='raised')
        show_button.image = show_photo
        show_button.grid(row=4, column=0, sticky='e', padx=(0, 124))
        hide_button = tk.Button(self, image=show_photo1, width=15,
height=15, relief='raised')
        hide_button.image = show_photo1

        loginButton = tk.Button(self, text="Σύνδεση",
                                command=lambda:
```

```
[self.login(controller, login_username, login_password)])

        def show_pass(passbox):
            if show_button.winfo_ismapped():
                show_button.grid_forget()
                hide_button.grid(row=4, column=0, sticky='e',
padx=(0, 124))
                passbox.config(show="")

        def hide_pass(passbox):
            if hide_button.winfo_ismapped():
                hide_button.grid_forget()
                show_button.grid(row=4, column=0, sticky='e',
padx=(0, 124))
                passbox.config(show="•")

        show_button.config(command=lambda: show_pass(password_entry))
        hide_button.config(command=lambda: hide_pass(password_entry))

        def log_bind(event):
            if login_entry.winfo_ismapped():
                loginButton.invoke()

        def back_bind(event):
            if login_entry.winfo_ismapped():
                button.invoke()

        button.bind('<Return>', back_bind)
        loginButton.bind('<Return>', log_bind)
        login_entry.bind('<Return>', log_bind)
        password_entry.bind('<Return>', log_bind)
        loginButton.grid(row=5, column=0)

        def check_button():
            if hide_button.winfo_ismapped():
                hide_button.grid_forget()
                show_button.grid(row=4, column=0, sticky='e',
padx=(0, 124))

        button.config(command=lambda: [check_button(),
controller.show_frame("AuthenticationForm"), text_clear(),
password_entry.config(show="•")])
        button.grid(row=6, column=0, pady=(30, 40))

        forget_label = tk.Label(self, text="Ξεχάσατε τον κωδικό
σας;", font=("Helvetica", 10, "bold"))
        forget_label.grid(row=7, column=0, pady=(0, 3))

        forget_button = tk.Button(self, text="Πατήστε εδώ",
command=lambda: self.controller.show_frame("ForgetPass"))
        forget_button.grid(row=8, column=0)


# Register form window
class RegisterForm(tk.Frame):
    # Method for sending OTP for registration
    @staticmethod
    def email_sent(register_email):
        digits = "0123456789"
        OTP = ""
        length = len(digits)
```

```
        for i in range(6):
            OTP += digits[math.floor(random.random() * length)]
        msg = OTP

        gmail_user = 'crud.otp@gmail.com'
        gmail_password = 'purcbrayvwbtfppi'

        register_email = StringVar(value=register_email)
        register_email = register_email.get()

        message = MIMEMultipart("alternative")
        message["Subject"] = 'Επιβεβαίωση εγγραφής στην εφαρμογή'
        message["From"] = formataddr((str(Header('CRUD', 'utf-8')),
'%s' % gmail_user))
        message["To"] = register_email
        html = """
<!DOCTYPE html>
<html lang="en" >
<head>
  <meta charset="UTF-8">

</head>
<body>
<!-- partial:index.partial.html -->
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
    <div style="text-align:center;">
      <p style="font-family:Arial Black; font-size:16pt;">
        Παρακαλώ πληκτρολογήστε τον παρακάτω κωδικό για να
ολοκληρωθεί η εγγραφή στην εφαρμογή</p>
      <p style="font-family:Arial Black; font-size:20pt;"> %s </p>
    </div>
</body>
</html>
<!-- partial -->
</body>
</html>
        """ % msg
        part1 = MIMEText(html, "html")
        message.attach(part1)
        try:
            smtp_server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
            smtp_server.ehlo()
            smtp_server.login(gmail_user, gmail_password)
            smtp_server.sendmail(gmail_user, register_email,
message.as_string())
            smtp_server.close()
            return msg
        except smtplib.SMTPRecipientsRefused as ex:
            print("Something went wrong….", ex)
            messagebox.showerror("Λάθος διεύθυνση Email", "Παρακαλώ
πληκτρολογήστε μια έγκυρη διεύθυνση Email.")

    # Method for registering account in the database
    def register(self, controller, register_email, register_name,
register_password, register_password_ver):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
```

```
            port=3306,
            db="ptixiaki",
            user="root",
            password="")
        cur = cnx.cursor()

        register_email = register_email.get()
        register_password = register_password.get()
        register_name = register_name.get()
        register_password_ver = register_password_ver.get()
        if str.strip(register_password_ver) != "" and
str.strip(register_password) != "" and str.strip(register_email) !=
"" \
                and str.strip(register_name) != "" and
register_password_ver == register_password\
                and len(register_password) > 7 and len(register_name)
<= 20 and re.fullmatch(regex, register_email):
            try:
                query = f"INSERT INTO users (name, email,
password,access_level) VALUES " + \
                        f"(%s,%s,%s,'0')"
                cur.execute(query, (register_name, register_email,
register_password))
                otp = self.email_sent(register_email)
                subprocess.call('python OTPVerification.py ' + otp +
' ' + '1' + ' ' + '&', shell=True)
                nl = ''
                if os.path.isfile("status.txt") is True:
                    f = open("status.txt", "r")
                    nl = f.read()
                    f.close()

                if nl == "('%s', True)" % otp:
                    cnx.commit()
                    messagebox.showinfo("Επιτυχής Εγγραφή", "Ο
λογαριασμός σας δημιουργήθηκε. Τώρα μπορείτε να συνδεθείτε στον
λογαριασμό σας.")
                    controller.show_frame("AuthenticationForm")
                    text_clear()
                else:
                    self.focus_force()
                    print("OTP was wrong")
            except (mysql.connector.Error, TypeError,
UnicodeEncodeError) as err:
                if err.__class__.__name__ == "IntegrityError" and
err.errno == 1062:
                    messagebox.showerror("Παρόμοια εισαγωγή", "Το
όνομα χρήστη ή το Email υπάρχει ήδη.")
                elif err.__class__.__name__ == "UnicodeEncodeError":
                    messagebox.showerror("Λάθος διεύθυνση email",
"Παρακαλώ ελέγξτε την διεύθυνση email που εισαγάγατε.")
                else:
                    print("Error: " + str(err))
            finally:
                if cnx.is_connected() is True:
                    cur.close()
                    cnx.close()
                    print('MySQL connection is closed')
                if os.path.isfile("status.txt") is True:
                    os.remove('status.txt')
        elif str.strip(register_email) == "" or
```

```
str.strip(register_name) == "" or str.strip(register_password) == ""
\
                or str.strip(register_password_ver) == "":
            messagebox.showerror("Υπάρχουν κενά πεδία", "Παρακαλώ
συμπληρώστε όλα τα πεδία.")
        elif len(register_password) <= 7:
            messagebox.showerror("Ελάχιστο όριο κωδικού",
                                 f"Παρακαλώ πληκτρολογήστε έναν
κωδικό με τουλάχιστον 8 χαρακτήρες.\nΧαρακτήρες:
{len(register_password)}.")
        elif len(register_name) > 20:
            messagebox.showerror("Μέγιστο όριο χαρακτήρων",
                                 f"Παρακαλώ πληκτρολογήστε ένα όνομα
χρήστη έως 20 χαρακτήρες.\nΧαρακτήρες: {len(register_name)}.")
        elif register_password_ver != register_password:
            messagebox.showerror("Ο κωδικός δεν είναι ίδιος",
"Παρακαλώ πληκτρολογήστε τον σωστό κωδικό.")
        elif not re.fullmatch(regex, register_email):
            messagebox.showerror("Λάθος διεύθυνση email", "Παρακαλώ
ελέγξτε την διεύθυνση email που εισαγάγατε.")

    # Constructor of this class
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        global email_box
        global username_box
        global password_box
        global password_ver_box
        label = tk.Label(self, text="Εγγραφή",
font=controller.title_font)
        label.grid(row=0, column=0, sticky="ew")

        register_email = StringVar()
        register_name = StringVar()
        register_password = StringVar()
        register_password_ver = StringVar()

        email_label = tk.Label(self, text="E-mail")
        email_label.grid(row=1, column=0)
        email_box = tk.Entry(self, width=40,
textvariable=register_email, justify="center")
        email_box.grid(row=2, column=0, pady=(0, 10))
        username_label = tk.Label(self, text="Όνομα χρήστη")
        username_label.grid(row=3, column=0)
        username_box = tk.Entry(self, width=30,
textvariable=register_name, justify="center")
        username_box.grid(row=4, column=0, pady=(0, 10))
        password_label = tk.Label(self, text="Κωδικός")
        password_label.grid(row=5, column=0)
        password_box = tk.Entry(self, width=30,
textvariable=register_password, show='•', justify="center")
        password_box.grid(row=6, column=0, pady=(0, 10))
        photo = Image.open("show-password-icon-18.jpg")
        photo1 = Image.open("show-password-icon-17.jpg")
        resized_photo = photo.resize((20,20), Image.ANTIALIAS)
        resized_photo1 = photo1.resize((20, 20), Image.ANTIALIAS)
        show_photo = ImageTk.PhotoImage(resized_photo)
        show_photo1 = ImageTk.PhotoImage(resized_photo1)
        show_button = tk.Button(self, image=show_photo, width=15,
height=15, relief='raised')
```

```
        show_button.image = show_photo
        show_button.grid(row=6, column=0, sticky='ne', padx=(0, 8))
        hide_button = tk.Button(self, image=show_photo1, width=15,
height=15, relief='raised')
        hide_button.image = show_photo1
        password_ver_label = tk.Label(self, text="Επανάληψη κωδικού")
        password_ver_label.grid(row=7, column=0)
        password_ver_box = tk.Entry(self, width=30,
textvariable=register_password_ver, show='•', justify="center")
        password_ver_box.grid(row=8, column=0, pady=(0, 10))
        show_button1 = tk.Button(self, image=show_photo, width=15,
height=15, relief='raised')
        show_button1.image = show_photo
        show_button1.grid(row=8, column=0, sticky='ne', padx=(0, 8))
        hide_button1 = tk.Button(self, image=show_photo1, width=15,
height=15, relief='raised')
        hide_button1.image = show_photo1
        register_button = tk.Button(self, text="Εγγραφή",
                                    command=lambda:
self.register(controller, register_email, register_name,

register_password, register_password_ver))
        register_button.grid(row=9, column=0, pady=(5, 30))

        def show_pass(passbox):
            if show_button.winfo_ismapped():
                show_button.grid_forget()
                hide_button.grid(row=6, column=0, sticky='ne',
padx=(0, 8))
                passbox.config(show="")

        def hide_pass(passbox):
            if hide_button.winfo_ismapped():
                hide_button.grid_forget()
                show_button.grid(row=6, column=0, sticky='ne',
padx=(0, 8))
                passbox.config(show="•")

        def show_pass1(passbox):
            if show_button1.winfo_ismapped():
                show_button1.grid_forget()
                hide_button1.grid(row=8, column=0, sticky='ne',
padx=(0, 8))
                passbox.config(show="")

        def hide_pass1(passbox):
            if hide_button1.winfo_ismapped():
                hide_button1.grid_forget()
                show_button1.grid(row=8, column=0, sticky='ne',
padx=(0, 8))
                passbox.config(show="•")

        show_button.config(command=lambda: show_pass(password_box))
        hide_button.config(command=lambda: hide_pass(password_box))
        show_button1.config(command=lambda:
show_pass1(password_ver_box))
        hide_button1.config(command=lambda:
hide_pass1(password_ver_box))

        def register_bind(event):
            if register_button.winfo_ismapped():
```

```
                register_button.invoke()

        def back_bind(event):
            if button.winfo_ismapped():
                button.invoke()

        register_button.bind('<Return>', register_bind)
        username_box.bind('<Return>', register_bind)
        password_box.bind('<Return>', register_bind)
        email_box.bind('<Return>', register_bind)
        password_ver_box.bind('<Return>', register_bind)

        def check_button():
            if hide_button.winfo_ismapped() or
hide_button1.winfo_ismapped():
                hide_button1.grid_forget()
                show_button1.grid(row=8, column=0, sticky='ne',
padx=(0, 8))
                hide_button.grid_forget()
                show_button.grid(row=6, column=0, sticky='ne',
padx=(0, 8))
                password_box.config(show="•")
                password_ver_box.config(show="•")

        button = tk.Button(self, text="Πίσω",
                           command=lambda: [check_button(),
controller.show_frame("AuthenticationForm"), text_clear()])
        button.grid(row=10, column=0)
        button.bind('<Return>', back_bind)


# Admin's window
class AdminPage(tk.Frame):
    # Constructor of this class
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        title_label = tk.Label(self, text="Παράθυρο διαχειριστή",
font=self.controller.title_font)
        title_label.grid(row=0, column=0, sticky="N", columnspan=2)
        log_out = tk.Button(self, text="Αποσύνδεση")
        log_out.config(command=lambda: [self.destroy_all_chk_box(),
self.destroy_recent_chk_box(), text_clear(),
self.destroy_all_frames(),

self.controller.show_frame("AuthenticationForm"),
                                        root.geometry("%dx%d+%d+%d" %
(width, height, app.winfo_screenwidth() / 2 - width / 2,

app.winfo_screenheight() / 2 - height / 2))])
        log_out.grid(row=0, column=1, sticky="e")
        self.pop3 = None
        self.pop4 = None

        def width_height():
            _width = int(root.winfo_screenwidth() / 2 - 1250 / 2)
            _height = int(root.winfo_screenheight() / 2 - 635 / 2)
            return _width, _height

        def dimensions_adminpage():
            admin_width = int(root.winfo_screenwidth() / 2 - 840 / 2)
```

```
            admin_height = int(root.winfo_screenheight() / 2 - 500 /
2)
            return admin_width, admin_height

        self.show_all_button = tk.Button(self, text="Όλα τα αρχεία",
width=20, height=2,
                                         command=lambda:
[controller.frames['ShowAllFiles'].back_button.

config(command=lambda: [self.controller.show_frame('AdminPage'),

controller.frames['ShowAllFiles'].user_clear_interior(),

controller.frames['ShowAllFiles'].all_clear_interior(),

root.geometry("840x405+%s+%s" % dimensions_adminpage()),

self.destroy_recent_chk_box(), self.recent_files_chk_box()]),

controller.frames['ShowAllFiles'].delete_button.grid(row=3, column=2,
sticky='e',

columnspan=1, ipadx=10, ipady=3,

padx=(0, 70)),

controller.frames['ShowAllFiles'].all_rename_button.grid(row=1,
column=2, sticky='w',

padx=(13, 0)),

controller.frames['ShowAllFiles'].description.config(state=tk.DISABLE
D),

self.controller.show_frame("ShowAllFiles"),

root.geometry("1250x635+%s+%s" % width_height()),

controller.frames['ShowAllFiles'].currentType.set('Όλα'),

controller.frames['ShowAllFiles'].currentType_user.set('Όλα'),

controller.frames['ShowAllFiles'].page_assigning_user(

controller.frames['ShowAllFiles'].show_all_user()),

controller.frames['ShowAllFiles'].user_page_fill(

controller.frames['ShowAllFiles'].show_all_user()),

controller.frames['ShowAllFiles'].page_assigning(

controller.frames['ShowAllFiles'].show_all()),

controller.frames['ShowAllFiles'].page_fill(controller.frames['ShowAl
lFiles'].show_all())])
        self.show_all_button.grid(row=4, column=1)
        self.upload_button = tk.Button(self, text="Ανέβασμα αρχείου",
width=20, height=2, command=lambda: self.upload())
        self.upload_button.grid(row=4, column=0, columnspan=2,
padx=(0, 140))
```

```
        # Method of the edit button that creates pop-up window
        # for editing user's credentials or deleting users
        def edit_users():
            pop3 = self.pop3 = tk.Toplevel(root)
            pop3.grab_set()
            pop3.focus_force()
            pop3.title("Επεξεργασία χρηστών")
            pop3.iconbitmap('uowm-logo-big.ico')
            width3 = pop3.winfo_screenwidth() / 2 - 327 / 2
            height3 = pop3.winfo_screenheight() / 2 - 100 / 2
            pop3.geometry("327x100+%s+%s" % (int(width3),
int(height3)))
            label = tk.Label(pop3, text="Διαλέξτε μια ενέργεια για
τους\n επιλεγμενους χρήστες", font=("Arial", 13))
            label.grid(row=0, column=0, padx=50, pady=10)

            def edit_user_info():
                if self.pop3.winfo_ismapped():
                    self.pop3.destroy()

                # creation of the pop-up window for editing user
credentials
                def edit_user(name, passw, email, _id):
                    _username = tk.StringVar()
                    _password = tk.StringVar()
                    _email = tk.StringVar()
                    pop4 = self.pop4 = tk.Toplevel(root)
                    pop4.grab_set()
                    pop4.focus_force()
                    pop4.title("Επεξεργασία πληροφοριών χρήστη")
                    pop4.iconbitmap('uowm-logo-big.ico')
                    width4 = pop4.winfo_screenwidth() / 2 - 400 / 2
                    height4 = pop4.winfo_screenheight() / 2 - 300 / 2
                    pop4.geometry("400x300+%s+%s" % (int(width4),
int(height4)))
                    pop4.resizable(False, False)
                    pop4_label = tk.Label(pop4, text="Στοιχεία
χρήστη", font=("Arial", 18, "bold"))
                    pop4_label.pack(padx=50, pady=10)
                    label_frame = tk.LabelFrame(pop4,
text="Στοιχεία", font=("Arial", 8, 'bold'), bd=3, width=140,
height=60)
                    label_frame.pack()
                    window_font = tkfont.Font(family="arial",
size=12, weight="bold")
                    window_font1 = tkfont.Font(family="arial",
size=11)
                    email_label = tk.Label(label_frame, text="Email",
font=window_font)
                    email_label.pack(pady=(5, 0), padx=5)
                    email_entry = tk.Entry(label_frame,
textvariable=_email, justify="center", font=window_font1)
                    email_entry.insert(0, email)
                    email_entry.pack(ipadx=50)
                    name_label = tk.Label(label_frame, text="Ονομα
χρήστη", font=window_font)
                    name_label.pack(pady=(5, 0), padx=5)
                    name_box = tk.Entry(label_frame,
textvariable=_username, justify="center", font=window_font1)
                    name_box.insert(0, name)
```

```
                    name_box.pack(ipadx=10)
                    password_label = tk.Label(label_frame,
text="Κωδικός", font=window_font)
                    password_label.pack(pady=(5, 0), padx=5)
                    pass_box = tk.Entry(label_frame,
textvariable=_password, justify="center", font=window_font1)
                    pass_box.insert(0, passw)
                    pass_box.pack(ipadx=10, pady=(0, 3))
                    update_button = tk.Button(label_frame,
text="Ενημέρωση στοιχείων")
                    update_button.pack(pady=(0, 3))

                    def count_characters(txt):
                        t = txt.get()
                        if len(t) > 20:
                            txt.delete(len(t)-1, 'end')

                    name_box.bind('<Any-KeyPress>', lambda e:
count_characters(name_box))
                    name_box.bind('<Any-KeyRelease>', lambda e:
count_characters(name_box), add='+')

                    # for the update button inside the edit user's
info window
                    def update_info(_username, _email, _password,
_id):
                        if name_box.get().strip() != '' and
email_entry.get().strip() != '' and pass_box.get().strip() != ''\
                                and "@" in email_entry.get() and "."
in email_entry.get() and len(pass_box.get()) > 7:
                            try:
                                cnx1 = mysql.connector.connect(
                                    host="127.0.0.1",
                                    port=3306,
                                    db="ptixiaki",
                                    user="root",
                                    password="")
                                cur1 = cnx.cursor()
                                _username1 = _username.get()
                                _password1 = _password.get()
                                _email1 = _email.get()
                                query1 = "UPDATE users SET name=%s,
password=%s, email=%s WHERE id=%s"
                                cur.execute(query1, (_username1,
_password1, _email1, _id))
                                cnx.commit()
                                messagebox.showinfo("Επιτυχής
ενημέρωση", "Η ενημέρωση των στοιχείων ήταν επιτυχής."

" Τώρα μπορείτε να κλείσετε το παράθυρο")
                            except mysql.connector.Error as err1:
                                print("MySQL error: {}".format(err1))
                            finally:
                                if cnx1.is_connected():
                                    cur1.close()
                                    cnx1.close()
                                    print("MySQL connection is
closed")
                        else:
                            if name_box.get().strip() == '' or
email_entry.get().strip() == '' or pass_box.get().strip() == '':
```

```
                                messagebox.showerror("Κενά πεδία",
"Δεν πρέπει να υπάρχει κανένα κενό πεδίο.")
                            elif len(pass_box.get()) <= 7:
                                messagebox.showerror("Ελάχιστο όριο
κωδικού",
                                                     f"Παρακαλώ
πληκτρολογήστε έναν κωδικό με τουλάχιστον 8 χαρακτήρες.\nΧαρακτήρες:"
                                                     f"
{len(pass_box.get())}.")
                            else:
                                messagebox.showerror("Λάθος email",
"Παρακαλώ πληκτρολογήστε μία έγκυρη διεύθυνση email.")
                        update_button.config(command=lambda:
update_info(_username, _email, _password, _id))

                    result = [var.get() for var in
self.user_check_box_identities_text if var.get()]
                    if len(result):
                        for res in result:
                            if res != '':
                                try:
                                    cnx = mysql.connector.connect(
                                        host="127.0.0.1",
                                        port=3306,
                                        db="ptixiaki",
                                        user="root",
                                        password="")
                                    cur = cnx.cursor()

                                    query = "SELECT name, password, email
FROM users WHERE id=%s"
                                    cur.execute(query, (res, ))
                                    user_info = cur.fetchone()
                                    user_name = user_info[0]
                                    user_email = user_info[2]
                                    user_password = user_info[1]
                                    edit_user(name=user_name,
email=user_email, passw=user_password, _id=res)
                                    root.wait_window(self.pop4)
                                    self.destroy_all_chk_box()

self.check_box_users(self.show_users())
                                except mysql.connector.Error as err:
                                    print("MySQL error: {}".format(err))
                                finally:
                                    if cnx.is_connected():
                                        cur.close()
                                        cnx.close()
                                        print("MySQL connection is
closed")

            button_edit = tk.Button(pop3, text="Επεξεργασία
στοιχείων", command=edit_user_info)
            button_edit.grid(row=1, column=0, sticky="w")
            button_delete = tk.Button(pop3, text="Διαγραφή
χρήστη/ων", command=lambda:
self.delete_users(self.user_check_box_identities_text))
            button_delete.grid(row=1, column=0, sticky="e")

        def check_users():
            result = [var.get() for var in
```

```
self.user_check_box_identities_text if var.get()]
            if len(result):
                edit_users()
            else:
                tk.messagebox.showerror("Δεν επιλέχθηκαν χρήστες",
"Παρακαλώ επιλέξτε τουλάχιστον έναν χρήστη για επεξεργασία.")

        def check_window(window):
            try:
                root.wait_window(window)
            except Exception as err:
                print("Error: {}".format(err))

        users_edit_button = tk.Button(self, text="Επεξεργασία
χρήστη/ων", command=lambda: [check_users(), check_window(self.pop3)])
        users_edit_button.grid(row=3, column=0, pady=(0, 10))
        files_delete = tk.Button(self, text="Διαγραφή",
command=lambda:
self.submit_delete(self.recent_check_box_identities_text))
        files_delete.grid(row=3, column=1, sticky='e', padx=(0, 55))
        download_files = tk.Button(self, text='Λήψη')
        download_files.config(command=lambda:
self.controller.frames["ShowAllFiles"].submit(self.recent_check_box_i
dentities_text))
        download_files.grid(row=3, column=1, sticky='e')
        user_search = StringVar()
        self.search_box_user = tk.Entry(self, width=25,
textvariable=user_search)

        def search_bind(button):
            if button.winfo_ismapped():
                button.invoke()

        photo = Image.open("309077_search_icon.png")
        resized_photo = photo.resize((20, 20), Image.ANTIALIAS)
        show_photo = ImageTk.PhotoImage(resized_photo)
        user_search_button = tk.Button(self, image=show_photo,
width=15, height=15, relief="raised")
        user_search_button.config(command=lambda:
[self.destroy_all_chk_box(), self.user_search(user_search)])
        user_search_button.image = show_photo
        self.search_box_user.bind('<Return>', lambda e:
search_bind(user_search_button))
        self.search_box_user.grid(row=1, column=0, sticky="W")
        user_search_button.grid(row=1, column=0, sticky="E")

    # Method that deletes selected users for the button delete users
    def delete_users(self, check_box):
        result = [var.get() for var in check_box if var.get()]
        if len(result):
            ask_delete = tk.messagebox.askyesno("Διαγραφή χρήστη/ων",
"Είστε σίγουροι ότι θέλετε να διαγράψετε τους επιλεγμένους χρήστες;")
            if ask_delete is True:
                for i in result:
                    self.delete_user_from_db(i)
                self.destroy_all_chk_box()
                self.check_box_users(self.show_users())
                if self.pop3.winfo_ismapped():
                    self.pop3.destroy()
            else:
                print("User not deleted")
```

```
        else:
            print("no users were selected")

    # Method that deletes a user from the database
    @staticmethod
    def delete_user_from_db(i):
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")
            cur = cnx.cursor()

            query = "DELETE FROM users WHERE users.id = %s;"
            cur.execute(query, (int(i),))
            cnx.commit()
        except mysql.connector.Error as err:
            print("MySQL Error: {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Shows all users inserted in the database
    @staticmethod
    def show_users():
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")
            cur = cnx.cursor()

            query = "SELECT id,name FROM users WHERE id <> %s"
            cur.execute(query, (int(row[0]),))
            result = cur.fetchall()
            return result
        except mysql.connector.Error as err:
            print("MySQL Error: {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Destroys the checkboxes used for users
    def destroy_all_chk_box(self):
        if len(self.user_check_box_identities):
            for ch in self.user_check_box_identities:
                ch.destroy()

    def destroy_all_frames(self):
        self.controller.destroy_frame()

    # Destroys the checkboxes used for 20 latest files
    def destroy_recent_chk_box(self):
```

```
        if len(self.recent_check_box_identities):
            for ch in self.recent_check_box_identities:
                ch.destroy()
        if len(self.recent_labels_identities):
            for l1, l2 in self.recent_labels_identities:
                l1.destroy()
                l2.destroy()

    # Creates the checkboxes for users
    def check_box_users(self, file_list):
        vscrollbar = tk.Scrollbar(self, orient='vertical')
        vscrollbar.grid(row=2, column=0, ipadx=2, ipady=99,
padx=(150, 0), sticky='wns')
        canvas = tk.Canvas(self, bd=1, highlightthickness=0,
yscrollcommand=vscrollbar.set)
        canvas.grid(row=2, column=0, sticky='w')
        vscrollbar.config(command=canvas.yview)
        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled
with it
        user_interior = tk.Listbox(canvas)
        interior_id = canvas.create_window(0, 0,
window=user_interior,
                                           anchor='nw')

        file_name_label = tk.Label(user_interior, text="Χρήστες",
width=20)
        file_name_label.grid(row=0, column=0, sticky='w')

        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner
frame
            size = (user_interior.winfo_reqwidth(),
user_interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=user_interior.winfo_reqwidth())

        user_interior.bind('<Configure>', _configure_interior)

        self.user_check_box_identities = []
        self.user_check_box_identities_text = []
        counter = 1
        count = 1
        if len(file_list) != 0:
            for rows in range(len(file_list)):
                label_text = file_list[rows]
                var = StringVar()
                check_box = tk.Checkbutton(user_interior,
variable=var, onvalue=label_text[0], offvalue='', text=label_text[1])
                check_box.deselect()
                self.user_check_box_identities.append(check_box)
                self.user_check_box_identities_text.append(var)
                check_box.grid(row=counter, column=0, sticky='w')
```

```
                counter += 1
                count += 1

        def _configure_canvas(event):
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id,
width=canvas.winfo_width())

        canvas.bind('<Configure>', _configure_canvas)

    # Shows the 20 recent files
    @staticmethod
    def show_recent():
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cur = cnx.cursor()
        try:
            query = "SELECT files.id , files.file_name , users.name,
files.date_added " \
                    "FROM files INNER JOIN file_types ON
files.file_type_id = file_types.id " \
                    "INNER JOIN users ON users.id = files.user_id " \
                    "ORDER BY files.date_added DESC, files.id DESC
LIMIT 20"
            cur.execute(query, ())
            rows = cur.fetchall()
            return rows
        except mysql.connector.Error as err:
            print("Error to the connection of MySQL {}".format(err))
        finally:
            if cnx.is_connected():
                cur.close()
                cnx.close()
                print("MySQL connection is closed")

    # Creates the checkboxes for 20 recent files
    def recent_files_chk_box(self):
        vscrollbar = tk.Scrollbar(self, orient='vertical')
        vscrollbar.grid(row=2, column=1, ipadx=2, ipady=99,
padx=(612, 0), sticky='wns')
        canvas = tk.Canvas(self, bd=1, highlightthickness=0,
yscrollcommand=vscrollbar.set)
        canvas.grid(row=2, column=1, sticky='w', padx=(30, 0))
        vscrollbar.config(command=canvas.yview)
        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled
with it
        user_interior = tk.Listbox(canvas)
        interior_id = canvas.create_window(0, 0,
window=user_interior,
                                           anchor='nw')
```

```
        file_name_label = tk.Label(user_interior, text="Πρόσφατα
αρχεία", width=40)
        file_name_label.grid(row=0, column=0, sticky='w')
        user_name_label = tk.Label(user_interior, text="Χρήστης",
width=20)
        user_name_label.grid(row=0, column=1)
        date_label = tk.Label(user_interior, text="Ημερομηνία",
width=20)
        date_label.grid(row=0, column=2)

        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner
frame
            size = (user_interior.winfo_reqwidth(),
user_interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=user_interior.winfo_reqwidth())

        user_interior.bind('<Configure>', _configure_interior)

        self.recent_check_box_identities = []
        self.recent_check_box_identities_text = []
        self.recent_labels_identities = []
        counter = 1
        count = 1
        file_list = self.show_recent()
        if len(file_list) != 0:
            for rows in range(len(file_list)):
                label_text = file_list[rows]
                var = StringVar()
                if len(label_text[1]) <= 50:
                    check_box = tk.Checkbutton(user_interior,
variable=var, onvalue=label_text[0], offvalue='', text=label_text[1])
                    check_box.deselect()
                else:
                    check_box = tk.Checkbutton(user_interior,
variable=var, onvalue=label_text[0], offvalue='',

text=label_text[1][:50] + "\n" + label_text[1][50:])
                    check_box.deselect()
                username_label = tk.Label(user_interior,
text=label_text[2])
                date_time_label = tk.Label(user_interior,
text=label_text[3])
                self.recent_check_box_identities.append(check_box)
                self.recent_check_box_identities_text.append(var)
                self.recent_labels_identities.append((username_label,
date_time_label))
                check_box.grid(row=counter, column=0, sticky='w')
                username_label.grid(row=counter, column=1)
                date_time_label.grid(row=counter, column=2)
                counter += 1
                count += 1

        def _configure_canvas(event):
```

```
            if user_interior.winfo_reqwidth() !=
canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id,
width=canvas.winfo_width())

        canvas.bind('<Configure>', _configure_canvas)

    # Deletes a file from database
    @staticmethod
    def delete_file(file_id):
        print("Starting to delete record")
        try:
            cnx = mysql.connector.connect(
                host="127.0.0.1",
                port=3306,
                db="ptixiaki",
                user="root",
                password="")

            cursor = cnx.cursor()

            query = "DELETE FROM files WHERE files.id = %s"
            cursor.execute(query, (file_id,))
            cnx.commit()
            print("The record has been deleted")
        except mysql.connector.Error as err:
            print("MySQL Error: {}".format(err))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Method that is called by the delete button to delete chosen
files
    def submit_delete(self, checkbox_identities):
        result = [var.get() for var in checkbox_identities if
var.get()]
        if len(result):
            ask_delete = tk.messagebox.askyesno("Διαγραφή αρχείων",
"Είστε σίγουροι ότι θέλετε να διαγράψετε τα επιλεγμένα αρχεία;")
            if ask_delete is True:
                with concurrent.futures.ThreadPoolExecutor() as
executor:
                    executor.map(self.delete_file, result)
                self.destroy_recent_chk_box()
                self.recent_files_chk_box()
                messagebox.showinfo("Επιτυχής διαγραφή", "Η διαγραφή
των επιλεγμένων αρχείων ήταν επιτυχής.")
            else:
                print("File not deleted")
        else:
            messagebox.showerror("Δεν επιλέχθηκαν αρχεία", "Παρακαλώ
επιλέξτε τουλάχιστον ένα αρχείο για διαγραφή.")
            print("no files were selected")

    # Choose the files to upload
    def upload(self):
        file_path = filedialog.askopenfilenames()
        self.files = list()
```

```
        self.description = list()
        for file in file_path:
            self.files.append(file)
        if len(self.files):
            question = messagebox.askyesnocancel("Περιγραφή
αρχείου/ων", "Πατήστε Yes, αν θέλετε να εισάγετε περιγραφή σε κάποιο
αρχείο\n"

"No, αν δεν θέλετε να εισαχθεί περιγραφή\nCancel, αν θέλετε να
ακυρώσετε το ανέβασμα"

" των αρχείων.")
            if question is True:
                for f in self.files:
                    self.description_pop(filename=f.rsplit('/', 1)[-
1])
                    root.wait_window(self.pop)
                self.upload_files()
            elif question is False:
                for _ in range(len(self.files)):
                    self.description.append("Κενό")
                self.upload_files()
        self.destroy_recent_chk_box()
        self.recent_files_chk_box()

    # Creates loading window for uploading files
    def upload_files(self):
        try:
            self.progress_step = float(100.0 / len(self.files))
            anim = None

            def pop():
                global pop2, progressbar2, files_label2
                pop2 = tk.Toplevel(root)
                pop2.geometry('700x380+%s+%s' %
(int(pop2.winfo_screenwidth() / 2 - 700 / 2),
int(pop2.winfo_screenheight() / 2 - 380 / 2)))
                pop2.resizable(False, False)
                style = ttk.Style()
                style.configure('TFrame', background='white')
                frame = ttk.Frame(pop2, style='TFrame')
                frame.pack()
                pop2.focus_force()
                pop2.grab_set()
                title_label = ttk.Label(frame, text="Παρακαλώ
περιμένετε μέχρι να ολοκληρωθεί η διεργασία",
font=self.controller.title_font)
                title_label.grid(row=0, column=0, pady=30)
                file = "Spinner.gif"

                info = Image.open(file)

                frames = info.n_frames  # gives total number of
frames that gif contains

                # creating list of PhotoImage objects for each frames
                im = [tk.PhotoImage(file=file, format=f"gif -index
{ia}") for ia in range(frames)]

                count = 0
                style.configure('TLabel', background='white')
```

```
                gif_label = ttk.Label(frame, image="",
style='TLabel')
                gif_label.grid(row=1, column=0, pady=(0, 10))
                file_label = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="Αρχεία:\n")
                file_label.grid(row=1, column=0, pady=(0, 18))
                files_label2 = ttk.Label(frame, style='TLabel',
font=("Arial", 14), text="1/1")
                files_label2.grid(row=1, column=0, pady=(8, 0))

                def animation(count):
                    global anim
                    im2 = im[count]

                    gif_label.configure(image=im2)
                    count += 1
                    if count == frames:
                        count = 0
                    anim = pop2.after(30, lambda: animation(count))

                animation(count)
                progressbar2 = ttk.Progressbar(frame,
mode='determinate', maximum=100, value=0)
                progressbar2.grid(row=2, column=0, pady=(0, 30),
ipadx=100, ipady=3)

            def setv(value):
                progressbar2["value"] += value

            def lp(bool):
                if bool:
                    pop2.update()

            def des():
                pop2.destroy()

            def run():
                count = 1
                for a in range(len(self.files)):
                    files_label2.config(text="%d/%d" % (count,
len(self.files)))
                    self.files_for_upload(self.files[a],
self.description[a])
                    setv(self.progress_step)
                    count += 1

            pop()
            t1 = Thread(target=run, daemon=True)
            t1.start()
            while True:
                lp(True)
                if t1.is_alive() is False:
                    lp(False)
                    break
            des()
        except (FileNotFoundError, ZeroDivisionError, Exception) as
err:
            print("File error {}".format(err))
        finally:
            self.destroy_recent_chk_box()
            self.recent_files_chk_box()
```

```
    # Uploads the files in the database
    def files_for_upload(self, file, description):
        filename = file.rsplit('/', 1)[-1]
        extension = filename.split('.', 1)[-1]
        date_time = "0-0-0 0:0:0"
        if extension.lower() in image_extensions:
            filetype_id = 2
            self.controller.frames['HomePage'].insertBLOB(filename,
file, int(row[0]), filetype_id, str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in text_extensions:
            filetype_id = 1
            self.controller.frames['HomePage'].insertBLOB(filename,
file, int(row[0]), filetype_id, str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in sound_extensions:
            filetype_id = 3
            self.controller.frames['HomePage'].insertBLOB(filename,
file, int(row[0]), filetype_id, str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        elif extension.lower() in video_extensions:
            filetype_id = 4
            self.controller.frames['HomePage'].insertBLOB(filename,
file, int(row[0]), filetype_id, str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        else:
            filetype_id = 5
            self.controller.frames['HomePage'].insertBLOB(filename,
file, int(row[0]), filetype_id, str(date_time))
            today = datetime.datetime.now()
            date_time = today.strftime("%Y-%m-%d %H:%M:%S")
        self.update_time(date_time)
        self.description_upload(filename, description)

    # Updates the time of uploading when the file has been
successfully uploaded
    @staticmethod
    def update_time(date_time):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "UPDATE files SET date_added = %s WHERE id =
(SELECT MAX(id) FROM files)"
            cursor.execute(query, (date_time, ))
            cnx.commit()
        except mysql.connector.Error as err:
            print("MySQL connection error: {}".format(err))
        finally:
            if cnx.is_connected():
                cursor.close()
```

```
                cnx.close()

    # Uploads description in database
    @staticmethod
    def description_upload(filename, description):
        cnx = mysql.connector.connect(
            host="127.0.0.1",
            port=3306,
            db="ptixiaki",
            user="root",
            password="")

        cursor = cnx.cursor()
        try:
            query = "INSERT INTO descriptions (files_id, description)
VALUES ((select max(id) from files where file_name = %s and user_id =
%s " \
                    "group by file_name), %s)"
            cursor.execute(query, (filename, int(row[0]),
description))
            cnx.commit()
        except mysql.connector.Error as error:
            print("Error connecting to MySQL {}".format(error))
        finally:
            if cnx.is_connected():
                cursor.close()
                cnx.close()
                print("MySQL connection is closed")

    # Creates pop-up window for writing descriptions for the files to
be uploaded
    def description_pop(self, filename):
        self.pop = tk.Toplevel(root)

        def on_close():
            if messagebox.askyesno("Κενή περιγραφή", "Αν κλείσετε το
παράθυρο η περιγραφή θα είναι κενή. Είστε σίγουροι ότι θέλετε να
συνεχίσετε;"):
                self.description.append("Κενό")
                self.pop.destroy()

        self.pop.protocol("WM_DELETE_WINDOW", on_close)
        self.pop.geometry('600x320+%s+%s' %
(int(self.pop.winfo_screenwidth() / 2 - 600 / 2),
int(self.pop.winfo_screenheight() / 2 - 320 / 2)))
        self.pop.resizable(False, False)
        self.pop.iconbitmap('uowm-logo-big.ico')
        style = ttk.Style()
        style.configure('TFrame', background='white')
        frame = ttk.Frame(self.pop, style='TFrame')
        frame.pack()
        self.pop.focus_force()
        root.grab_release()
        self.pop.grab_set()
        label = tk.Label(frame, text="Εισάγωγη περιγραφής",
font=self.controller.title_font)
        label.grid(row=0, column=0)
        file_label = tk.Label(frame, text="Όνομα αρχείου: %s" %
filename, font=("Arial", 13))
        file_label.grid(row=1, column=0)
        text = scrolledtext.ScrolledText(frame, height=10, width=50,
```

```
spacing1=1, spacing3=1)
        text.grid(row=2, column=0)
        count_label = tk.Label(frame, text="Χαρακτήρες: 0/256")
        count_label.grid(row=3, column=0, padx=(0, 100), sticky="e")

        def count_characters(txt):
            t = txt.get(1.0, tk.END)
            if len(t) > 257:
                txt.delete('%s - 2c' % 'end')

        text.bind('<Any-KeyPress>',
                  lambda e: [count_characters(text),
count_label.config(text="Χαρακτήρες: %s/256" % (len(text.get(1.0,
tk.END)) - 1))])
        text.bind('<Any-KeyRelease>',
                  lambda e: [count_characters(text),
count_label.config(text="Χαρακτήρες: %s/256" % (len(text.get(1.0,
tk.END)) - 1))],
                  add='+')

        def text_is_empty(txt):
            t = txt.get(1.0, '%s - 1c' % 'end')
            if t.strip() == '':
                self.description.append("Κενό")
            else:
                self.description.append(t)

        save = tk.Button(frame, text="Αποθήκευση", command=lambda:
[text_is_empty(text), self.pop.destroy()], height=1)
        save.grid(row=3, column=0, sticky="e")

    # Method for the search field
    def user_search(self, user_search):
        file_list = self.show_users()
        user_search = user_search.get()
        if user_search == '' or str.isspace(user_search) is True:
            self.destroy_all_chk_box()
            self.check_box_users(file_list)
        else:
            user_files = []
            for file in file_list:
                if user_search in file[1]:
                    user_files.append(file)
                else:
                    continue
            self.destroy_all_chk_box()
            self.check_box_users(user_files)


# Window for the forgot password window
class ForgetPass(tk.Frame):
    # Constructor for this class
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        title_label = tk.Label(self, text="Παρακαλώ εισάγετε το email
σας για να\nαρχίσει η διαδικασία αλλαγής κωδικού",
                               font=self.controller.title_font)
        title_label.pack(pady=(0, 30))

        given_email = tk.StringVar()
```

```
        self.email_entry = tk.Entry(self, textvariable=given_email,
width=50, font=("Helvetica", 10), justify="center")
        self.email_entry.pack(pady=(0, 5), ipady=4)

        def send_bind(event):
            if send_button.winfo_ismapped():
                send_button.invoke()
        self.email_entry.bind('<Return>', send_bind)

        send_button = tk.Button(self, text="Συνέχεια",
command=lambda:  self.email_veri(self.email_sent(given_email.get())))
        send_button.pack(pady=(0, 30))
        send_button.bind('<Return>', send_bind)

        back_button = tk.Button(self, text="Πισω", command=lambda:
[self.controller.show_frame("LoginForm"), self.clear_text()])
        back_button.pack()

    # Clears the email field
    def clear_text(self):
        self.email_entry.delete(0, tk.END)

    # Checks if the email exists in the Database and sents
    # an OTP in the email for changing password
    def email_sent(self, written_email):
        def check_if_exists(written1_email):
            try:
                cnx = mysql.connector.connect(
                    host="127.0.0.1",
                    port=3306,
                    db="ptixiaki",
                    user="root",
                    password="")
                cur = cnx.cursor()
                written1_email = StringVar(value=written1_email)
                given_email = written1_email.get()
                query = "SELECT id,name FROM users WHERE email = %s"
                cur.execute(query, (given_email,))
                result = cur.fetchall()
                if len(result):
                    return True
                else:
                    messagebox.showerror("Μη καταχωρημένο email", "Το
email που έχετε εισάγει δεν είναι καταχωρημένο στο σύστημα."

"Παρακαλώ εισάγετε το email που χρησιμοποιείτε για να συνδεθείτε στο
λογαριαμό σας")
                    return False
            except mysql.connector.Error as err:
                print("MySQL Error: {}".format(err))
            finally:
                if cnx.is_connected():
                    cur.close()
                    cnx.close()
                    print("MySQL connection is closed")

        if check_if_exists(written_email):
            digits = "0123456789"
            OTP = ""
            length = len(digits)
            for i in range(6):
```

```
                OTP += digits[math.floor(random.random() * length)]
            msg = OTP

            gmail_user = 'crud.otp@gmail.com'
            gmail_password = 'purcbrayvwbtfppi'

            written_email = StringVar(value=written_email)
            _written_email = written_email.get()
            message = MIMEMultipart("alternative")
            message["Subject"] = 'Αλλαγή κωδικού πρόσβασης'
            message["From"] = formataddr((str(Header('CRUD', 'utf-
8')), '%s' % gmail_user))
            message["To"] = _written_email
            html = """
            <!DOCTYPE html>
            <html lang="en" >
            <head>
              <meta charset="UTF-8">

            </head>
            <body>
            <!-- partial:index.partial.html -->
            <html>
            <head>
              <meta charset="UTF-8">
            </head>
            <body>
                <div style="text-align:center;">
                  <p style="font-family:Arial Black; font-
size:16pt;">
                    Παρακαλώ πληκτρολογήστε τον παρακάτω κωδικό για
να προχωρήσει η διαδικαία αλλαγής κωδικού στην εφαρμογή</p>
                  <p style="font-family:Arial Black; font-
size:20pt;"> %s </p>
                </div>
            </body>
            </html>
            <!-- partial -->
            </body>
            </html>
            """ % msg
            part1 = MIMEText(html, "html")
            message.attach(part1)
            try:
                smtp_server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
                smtp_server.ehlo()
                smtp_server.login(gmail_user, gmail_password)
                smtp_server.sendmail(gmail_user, _written_email,
message.as_string())
                smtp_server.close()
                print("Email sent successfully!")
                self.get_email(_written_email)
                return msg
            except (smtplib.SMTPRecipientsRefused, socket.gaierror)
as ex:
                if ex.__class__.__name__ == "socket.gaierror" and
ex.errno == 11001:
                    print("Something went wrong….", ex)
                    messagebox.showerror("Πρόβλημα επικοινωνίας",
"Υπάρχει ένα πρόβλημα επικοινωνίας με τον διακομιστή του email.
Παρακαλώ "
```

```
"προσπαθήστε ξανά σε λίγο")
                else:
                    print("Something went wrong….", ex)
                    messagebox.showerror("Λάθος διεύθυνση Email",
"Παρακαλώ πληκτρολογήστε μια έγκυρη διεύθυνση Email.")

    # Shows pop-up window for inserting OTP
    def email_veri(self, msg):
        subprocess.call('python OTPVerification.py ' + msg + ' ' +
'2' + ' ' + '&', shell=True)
        nl = ''
        if os.path.isfile("status.txt") is True:
            f = open("status.txt", "r")
            nl = f.read()
            f.close()
        print("OTP verification result: {}".format(nl))
        if nl == "('%s', True)" % msg:
            self.controller.show_frame("ChangeCode")
            self.clear_text()
        else:
            print('OTP was wrong')

    def get_email(self, email1):
        self.email1 = email1


# Shows window for changing password
class ChangeCode(tk.Frame):
    # Constructor for this class
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        title_label = tk.Label(self, text="Αλλαγή κωδικού πρόσβασης",
font=self.controller.title_font)
        title_label.pack(pady=(0, 30))

        def accept_bind(event):
            if accept.winfo_ismapped():
                accept.invoke()

        pass_label = tk.Label(self, text="Νέος κωδικός πρόσβασης",
font=(10))
        pass_label.pack(pady=(0, 5))
        given_pass = tk.StringVar()
        self.pass_entry = tk.Entry(self, textvariable=given_pass,
width=30, font=("Helvetica", 10), justify="center")
        self.pass_entry.pack(pady=(0, 10), ipady=4)
        self.pass_entry.bind('<Return>', accept_bind)

        pass_ver_label = tk.Label(self, text="Επανάληψη κωδικού
πρόσβασης", font=(10))
        pass_ver_label.pack(pady=(0, 5))
        given_pass_ver = tk.StringVar()
        self.pass_ver_entry = tk.Entry(self,
textvariable=given_pass_ver, width=30, font=("Helvetica", 10),
justify="center")
        self.pass_ver_entry.pack(pady=(0, 5), ipady=4)
        self.pass_ver_entry.bind('<Return>', accept_bind)

        accept = tk.Button(self, text="Αλλαγή κωδικού")
```

```
        accept.config(command=lambda:
self.change_code(given_pass.get(), given_pass_ver.get(),
self.controller.frames["ForgetPass"].email1))
        accept.pack(pady=(0, 30))
        accept.bind('<Return>', accept_bind)

        def cancel_bind(event):
            if cancel.winfo_ismapped():
                cancel.invoke()

        cancel = tk.Button(self, text="Ακύρωση", command=lambda:
[self.clear_text(), self.controller.show_frame("LoginForm")])
        cancel.pack()
        cancel.bind('<Return>', cancel_bind)

    def clear_text(self):
        self.pass_entry.delete(0, tk.END)
        self.pass_ver_entry.delete(0, tk.END)

    # Checks if the password and the repeated password are the same
and if they
    # the password is being updated
    def change_code(self, code, code_ver, email1):
        if code == code_ver and len(code) > 7:
            try:
                cnx = mysql.connector.connect(
                    host="127.0.0.1",
                    port=3306,
                    db="ptixiaki",
                    user="root",
                    password="")
                cur = cnx.cursor()
                code = StringVar(value=code)
                _code = code.get()
                email1 = StringVar(value=email1)
                _email1 = email1.get()
                query = "UPDATE users SET password = %s WHERE email =
%s"
                cur.execute(query, (_code, _email1))
                cnx.commit()
                messagebox.showinfo("Επιτυχής αλλαγή κωδικού",
                                    "Ο κωδικός σας άλλαξε με
επιτυχία. Τώρα μπορείτε να συνδεθείτε στον λογαριασμό σας με τον "
                                    "καινούργιο κωδικό.")
                self.controller.show_frame("LoginForm")
                self.clear_text()
            except mysql.connector.Error as err:
                print("MySQL Error: {}".format(err))
            finally:
                if cnx.is_connected():
                    cur.close()
                    cnx.close()
                    print("MySQL connection is closed")
        elif len(code) <= 7:
            messagebox.showerror("Ελάχιστο όριο κωδικού",
                                 f"Παρακαλώ πληκτρολογήστε έναν
κωδικό με τουλάχιστον 8 χαρακτήρες.\nΧαρακτήρες:
{len(register_password)}.")
        else:
            messagebox.showerror("Λανθασμένος κωδικός", "Ο κωδικός
δεν είναι ίδιος. Παρακαλώ ελέγξτε τον κωδικό σας ξανά.")
```

```
# Clears some fields
def text_clear():
    email_box.delete(0, tk.END)
    username_box.delete(0, tk.END)
    password_box.delete(0, tk.END)
    password_ver_box.delete(0, tk.END)
    login_entry.delete(0, tk.END)
    password_entry.delete(0, tk.END)


# Main for the program
if __name__ == "__main__":
    global root
    app = SampleApp()
    root = app
    app.tk_setPalette(background='white')
    app.title("CRUD")
    app.iconbitmap('uowm-logo-big.ico')
    width = int(app.winfo_screenwidth() * 0.44)
    height = int(app.winfo_screenheight()) * 0.53
    app.geometry("%dx%d+%d+%d" % (width, height,
app.winfo_screenwidth() / 2 - width / 2, app.winfo_screenheight() / 2
- height / 2))
    root.mainloop()
```