ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Σύστημα παρακολούθησης πρακτικών ασκήσεων

Πτυχιακή εργασία

Παναγιωτόπουλος Ιωάννης

Αθήνα , 2022

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATICS AND TELEMATICS

Internship tracker system

Bachelor thesis

Panagiotopoulos Ioannis

Athens , 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Θωμάς Καμαλάκης
Καθηγητής , Πληροφορικής και Τηλεματικής , Χαροκόπειο Πανεπιστήμιο Αθηνών

Ηρακλής Βαρλάμης
Αναπληρωτής καθηγητής , Πληροφορικής και Τηλεματικής , Χαροκόπειο Πανεπιστήμιο Αθηνών

Ανάργυρος Τσαδήμας
Μέλος Ε.Δ.Ι.Π. , Πληροφορικής και Τηλεματικής , Χαροκόπειο Πανεπιστήμιο Αθηνών

Ο Παναγιωτόπουλος Ιωάννης

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί , χωρίς να αλλάξει το περιεχόμενο της εργασίας μου , να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της , να την αντιγράψει σε οποιοδήποτε μέσο ή/ και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Η εκπόνηση της παρούσας πτυχιακή εργασίας έλαβε χώρα στο στο Τμήμα Πληροφορικής και Τηλεματικής , της Σχολής Ψηφιακής Τεχνολογίας , του Χαροκοπείου Πανεπιστημίου , υπό την επίβλεψη του Καθηγητή , και νυν προέδρου του τμήματος Θωμά Καμαλάκη.

Καταρχήν θα ήθελα να ευχαριστήσω τον κύριο Θωμά Καμαλάκη , ο οποίος ήταν ο επιβλέπων καθηγητής στην εργασία μου. Καθόλη τη διάρκεια της περάτωσης της πτυχιακής μου εργασίας , υπήρξε σημαντικός αρωγός σε αυτή μου τη προσπάθεια να συνεισφέρω το έργο μου στο πανεπιστήμιο. Εκτιμώ την προθυμία αλλά και το ενδιαφέρον που επέδειξε κατά το χρονικό διάστημα που συνεργαστήκαμε και το γεγονός πως σε κάθε πρόβλημα που είχα να αντιμετωπίσω η ανταπόκριση του ήταν άμεση και η βοήθεια του επαρκής. Πιστεύω πως μέσω της προσπάθειας μου να εκπονήσω αυτή την εργασία , αποκόμισα πολλά οφέλη που σχετίζονται διεύρυνση των τεχνικών γνώσεων και την ανάπτυξη τεχνολογικών δεξιοτήτων αλλά και χρησιμοποίησα τη κριτική μου σκέψη σε προβλήματα που βρίσκονται στον πραγματικό κόσμο , και πιο συγκεκριμένα στο πανεπιστήμιο έπειτα από αρκετές συναντήσεις που είχαμε για τη διαμόρφωση των τελικών απαιτήσεων και σχεδίασης του συστήματος. Επίσης θα ήθελα να ευχαριστήσω τα μέλη της εξεταστικής επιτροπής , κύριο Ανάργυρο Τσαδήμα και Ηρακλή Βαρλάμη για το διδακτικό τους έργο στο Πανεπιστήμιο και για τις σημαντικές γνώσεις που μου μετέδωσαν κατά τη διάρκεια των σπουδών μου.

Ευγνωμονώ σε βάθος την οικογένεια μου , για την ηθική τους στήριξη κατα τη διάρκεια των σπουδών μου , τη συμπαράσταση τους σε κάθε δυσκολία που αντιμετώπισα και την αφοσίωση τους ενόσω ήμουν προσηλωμένος στο στόχο μου. Επίσης , θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου Ιωάννη Μαυρομματάκη και Θεοφάνη Παπάζογλου για όλα όσα περάσαμε κατά τα φοιτητικά μας χρόνια αλλά και για τις συνεργασίες μας εντός της σχολής που με βοήθησαν σε πολλούς τομείς της ακαδημαϊκής ζωής μου.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά

Στη σήμερον εποχή η τεχνολογία έχει διαδραματίσει κομβικό ρόλο στην εξέλιξη της κοινωνίας μας και αποτελεί αναπόσπαστο κομμάτι της καθημερινότητας μας. Η τεχνολογία συνιστά μέσο ένωσης και δημιουργικής σύνθεσης της πνευματικής όψης του πολιτισμού(έρευνα, πολιτική, επιστήμη, παιδεία) αλλά και της υλικής(τεχνογνωσία , μέθοδοι παραγωγής).Η πρακτική αξία που μας δίνεται από τη χρήση της πληροφορικής και των υπολογιστών, έγκειται στο γεγονός πως ποικίλες διαδικασίες σε διάφορες εκφάνσεις της ζωής μας έχουν απλουστευθεί και ο χρόνος διεκπεραίωσης τους έχει μειωθεί σημαντικά συγκριτικά με το παρελθόν .Στον επιχειρηματικό τομέα και όχι μόνο , παρατηρούμε πως πολλοί οργανισμοί διαθέτουν πληροφοριακά συστήματα για τη διαχείριση αλλά και σε μερικές περιπτώσεις την διάθεση των υπηρεσιών τους. Εμμέσως πλην σαφώς , με τη στοχευμένη χρήση της τεχνολογίας έχει επιτευχθεί η αυτοματοποίηση των διαδικασιών οι οποίες απαιτούν χρόνο στους πελάτες και τους αρμόδιους.
Λαμβάνοντας υπόψη τα παραπάνω , θέλησα και εγώ με τη σειρά μου να ασχοληθώ με την ανάπτυξη ενός πληροφοριακού συστήματος στα πλαίσια των δραστηριοτήτων του πανεπιστημίου , με σκοπό να αξιοποιήσω τις γνώσεις και τα εφόδια που μου δόθηκαν κατά τη διάρκεια των σπουδών μου ώστε με τη χρήση της πληροφορικής να λύσω ζητήματα τα οποία αφορούν την ακαδημαϊκή ζωή των μελών του ιδρύματος. Κινητήριος δύναμη για την ενασχόληση μου με αυτό το έργο , ήταν η επιθυμία μου να λύσω ένα ουσιαστικό πρόβλημα που απαντάται στο παραγωγικό τομέα της σχολής μου και κατ' επέκταση του πανεπιστημίου , αλλά και μέσω της εργασίας μου να αναπτύξω περαιτέρω τις τεχνικές μου δεξιότητες οι οποίες θα με βοηθήσουν στο μελλοντικό μου βίο.

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη ενός συστήματος για την παρακολούθηση των πρακτικών ασκήσεων του Πανεπιστημίου. Σε αυτό το έργο αποπειράθηκα να αναπτύξω ένα λογισμικό , πιο συγκεκριμένα ένα πληροφοριακό σύστημα το οποίο θα έκανε τη διαχείριση των αιτήσεων πρακτικής άσκησης πιο εύκολη για όλα τα εμπλεκόμενα μέλη(Γραμματεία , Φοιτητές , Φορείς Π.Α. , Υπεύθυνους Καθηγητές) .

Εν κατακλείδι , το εν λόγω σύστημα "έρχεται" να δώσει ένα κοινό κανάλι ανταλλαγής πληροφοριών σχετικά με τις πρακτικές ασκήσεις , ούτως ώστε οι φοιτητές και φορείς πρακτικής άσκησης να μη χρειάζεται να προσκομίζουν δικαιολογητικά με μια ή περισσότερους μεθόδους επικοινωνίας , αλλά μέσω μιας έμπιστης πλατφόρμας του πανεπιστημίου η οποία διαθέτει ένα προτυποποιημένο

τρόπο για τη προώθηση των δικαιολογητικών και τα αιτήματα τους καθώς και τον έλεγχο της εγκυρότητας τους.

Λέξεις κλειδιά: Πληροφοριακό Σύστημα , Διαχείριση αιτήσεων , Πρακτική άσκηση

Abstract ή Περίληψη στα Αγγλικά

Nowadays technology has played a vital role in the evolution of our society and it is an indivisible part of our everyday lives. Technology is a means of connection and creative composition of the spiritual dimension of our civilisation and its material one.The substantial value , which derives from the use of technology lies in the fact that some processes have been simplified in many aspects of our lives and the time required for their completion has been reduced significantly in contrast to the past.Speaking of business sector and not only that , we are observing that a significant number of organizations use Information Systems for the management and sometimes for providing their services.Indirectly but clearly , with the targeted use of technology enables the automation of various procedures which require time for the customers and the people in charge of the operations.

Taking into consideration all the above , I decided by myself to be actively engaged in the development of an Information System in the context of our university's activities, with ultimate goal to utilize my knowledge and skills which were acquired during my studies in order to solve real world problems regarding the lives of our academic members on this institution.The main driving force for my involvement on this project was mainly due to my eagerness and strong will to solve substantial problems which are encountered in the productive sector of my department and university , and the chance to enhance further my technical skills which will be a key factor in my future life.

The main objective of the current thesis is the development of a System for tracking student internships on the University.On this project , I conducted an effort to produce a certain kind of software which would allow us to manage the internships' applications in an easier manner for all the involved members(Secretarians , Students, Internship Carriers , Supervising Professors).

To conclude , the said system will provide us with a single channel for exchanging information regarding the internships , which would practically resort to not having to retrieve their documents with one or more means of communication , but instead

through a trusted platform which is powered by the University which mandates a standard way for the forwarding of applications and the requests along with the inspections for the validity of each request.

Keywords: Information System , Application Management , Internships

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

HTML	Hypertext Markup Language
CSS	Cascade Style Sheets
Π.Α.	Πρακτική Άσκηση
Β.Δ.	Βάση Δεδομένων
ΣΧ.Β.Δ	Σχεσιακή Βάση Δεδομένων
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
MVT	Model View Template
MVC	Model View Controller
Η/Υ	Ηλεκτρονικός Υπολογιστής.
CRUD	Create Retrieve Update Delete
LDAP	Lightweight Directory Access Protocol
UI	User Interface
ORM	Object Relational Mapping
SSL	Secure Sockets Layer

HTML	Hypertext Markup Language
CSS	Cascade Style Sheets
Π.Α.	Πρακτική Άσκηση
Β.Δ.	Βάση Δεδομένων
ΣΧ.Β.Δ	Σχεσιακή Βάση Δεδομένων
HTTPS	Hypertext Transfer Protocol Secure
SQL	Structured Query Language
UUID	Universally Unique Identifier
JSON	Javascript Object Notation
XML	Extensible Markup Language
GIS	Geographic Information Systems
CBV	Class Based View
FQDN	Fully Qualified Domain Name
Λ.Σ.	Λειτουργικά Συστήματα
URL	Uniform Resource Locator
REST	Representational state transfer

ΚΕΦ.1 Εισαγωγή

1.1 Στόχος της εργασίας

Ο στόχος της παρούσας πτυχιακής εργασίας είναι η υλοποίηση ενός πληροφοριακού συστήματος για τη παρακολούθηση των πρακτικών ασκήσεων σε όλα τα τμήματα του Χαροκοπείου Πανεπιστημίου για τα προπτυχιακά προγράμματα σπουδών.

Με το εξής σύστημα διευκολύνουμε τους αιτούντες να υποβάλλουν τις αιτήσεις τους με τα απαραίτητα δικαιολογητικά σε μια ασφαλή και έμπιστη πλατφόρμα που υποστηρίζεται από το Πανεπιστήμιο , δίχως η επικοινωνία να γίνεται με ποικίλα κανάλια και τρόπους , ομοιοτρόπως και για την ενημέρωση σχετικά με τις φάσεις παρακολούθησης της πρακτικής άσκησης. Οι εμπλεκόμενοι τελικοί χρήστες της υπηρεσίας αυτής είναι οι φοιτητές και οι φορείς της πρακτικής άσκησης , οι οποίοι υποβάλλουν τη συμμετοχή τους στο πρόγραμμα της Π.Α. του πανεπιστημίου για το τρέχον ακαδημαϊκό έτος. Επιπροσθέτως , αυτό το πληροφοριακό σύστημα εμπλέκει τους υπεύθυνους καθηγητές των Π.Α. και τη γραμματεία , οι οποίοι συμβάλλουν σε διάφορες διαδικασίες της διεξαγωγής της Π.Α.

Οι φοιτητές έχουν τη δυνατότητα να εγγραφούν στο σύστημα , να δηλώσουν τη πρόθεση τους να συμμετάσχουν στο πρόγραμμα Π.Α κατά το τρέχον έτος , να δουν τις διαθέσιμες θέσεις εργασίας που έχουν αναρτήσει οι εγκεκριμένοι φορείς , να επιλέξουν τις αντίστοιχες θέσεις εργασίας που επιθυμούν , να ενημερωθούν για την ανάθεση τους , να υποβάλλουν τα δικαιολογητικά που χρειάζονται για την αξιολόγηση της πρακτικής τους και να ενημερωθούν για τη τελική βαθμολογία τους.

Οι φορείς μέσω των εκπροσώπων τους δύναται να υποβάλλουν αίτημα εισαγωγής στο πρόγραμμα της Π.Α. , να αναρτούν τις θέσεις εργασίας που ζητούν για αυτή τη περίοδο , να επιλέγουν φοιτητές προς κάλυψη αυτών των θέσεων σε συνεργασία με τη γραμματεία μέσω της πλατφόρμας και να αξιολογούν τις πρακτικές των φοιτητών που εκπλήρωσαν το χρέος της Π.Α. στο φορέα τους.

Οι καθηγητές καθόλη τη διάρκεια του κύκλου του συστήματος μένουν ενήμεροι για τις νέες θέσεις Π.Α. που προκύπτουν στις οποίες είναι επιβλέποντες , βλέπουν τις τελικές αναθέσεις των φοιτητών που συμμετέχουν σε θέσεις πρακτικής άσκησης

στις οποίες είναι υπεύθυνοι και βαθμολογούν τις πρακτικές των φοιτητών που έχουν αναλάβει.

Οι γραμματείες των τμημάτων του πανεπιστημίου έχουν την εποπτεία όλων των αιτήσεων στο σύστημα. Κάποιες από τις αρμοδιότητες τους είναι η έγκριση νέων φορέων Π.Α. στο σύστημα , έγκριση φοιτητών που επιθυμούν να συμμετάσχουν στη πρακτική άσκηση με βάση τα κριτήρια που ορίζονται για τη συμμετοχή σε Π.Α. (αριθμός χρωστούμενων μαθημάτων , έτος φοίτησης) , έγκριση αναρτημένων θέσεων εργασιών από τους φορείς , πρόταση αναθέσεων για Π.Α. σε φοιτητές , έλεγχος δικαιολογητικών αξιολόγησης Π.Α. φοιτητών , φορέων και καθηγητών.

Θεωρούμε πως η υλοποίηση ενός τέτοιου πληροφοριακού συστήματος διευκολύνει σε μεγάλο βαθμό τη διεξαγωγή των Π.Α. καθόλη τη διάρκεια τους , από την ένταξη των εμπλεκόμενων στο πρόγραμμα έως την ανακοίνωση των τελικών αναθέσεων των Π.Α. και την αξιολόγηση των φορέων και των φοιτητών. Ο κύκλος ζωής ενός αιτήματος εκτυλίσσεται εντός του συστήματος και όλα τα εμπλεκόμενα μέλη έχουν ως μοναδικό σημείο αναφοράς το σύστημα , ώστε να ενημερώνονται για την εξέλιξη ενός αιτήματος και να λαμβάνουν τις κατάλληλες ενέργειες εντός της πλατφόρμας. Η κεντρικοποίηση των δεδομένων σχετικά με την Π.Α. συνιστά θετικό παράγοντα για τη διεξαγωγή της , διότι τα δεδομένα είναι ευκόλως προσβάσιμα και έτσι ελαττώνονται τα περιθώρια για λάθη όταν διενεργούνται διαδικασίες κατά τις οποίες λαμβάνονται υπόψη. Επίσης η μείωση της γραφειοκρατίας είναι γεγονός , εφόσον οι διαδικασίες για την διεξαγωγή των φάσεων της Π.Α. είναι πλήρως ψηφιοποιημένες με αρκετές διευκολύνσεις που παρέχονται από το γραφικό περιβάλλον της web εφαρμογής μας.

Με τις υφιστάμενες διαδικασίες που συναντάμε σήμερα στο πανεπιστήμιο όσον αφορά τη διαχείριση των αιτήσεων Π.Α , διαπιστώνουμε τα εξής μειονεκτήματα:

1. Η εισαγωγή νέων φορέων στο πρόγραμμα της Π.Α απαιτεί αρκετές γραφειοκρατικές διαδικασίες και πολλαπλές επικοινωνίες για την επικύρωση των στοιχείων τους.

2. Για το αρχείο , συχνά συναντάμε μεθόδους αποθήκευσης όπως spreadsheets (λογιστικά φύλλα) σε Microsoft Excel , LibreOffice και πολλά άλλα προγράμματα. Τα δεδομένα μπορούν εύκολα να τροποποιηθούν με μη επιθυμητό τρόπο , και πολλές φορές να χαθούν λόγω απροσεξίας ή κακής συνεννόησης μεταξύ των αρμόδιων υπαλλήλων. Συχνά κατά το πέρασμα των στοιχείων μπορούν να γίνουν τυπογραφικά λάθη σε ευαίσθητα δεδομένα.

3. Η ενημέρωση για την εισαγωγή νέων θέσεων εργασίας από τους φορείς ενδέχεται να είναι αρκετά συχνή , οπότε οι ενδεχόμενες επικοινωνίες αυξάνονται με αποτέλεσμα να προστίθεται μεγαλύτερος όγκος εργασίας για τις γραμματείες.

4. Τα περιθώρια λάθους κατά τη προσκόμιση των δικαιολογητικών από τη μεριά του φοιτητή αυξάνονται όταν δεν υπάρχουν προτυποποιημένοι

τρόποι για τη προώθηση τους στη γραμματεία. Ως αποτέλεσμα έχουμε συχνότερες επικοινωνίες της γραμματείας για τη σωστή υποβολή της αίτησης και πολλές φορές ίσως υπάρξουν σημαντικά λάθη στη καταχώρηση του αρχείου.

5. Η ενημέρωση σχετικά με τη προσθήκη νέων αναθέσεων και θέσεων εργασίας απαιτεί συχνές επικοινωνίες μεταξύ της γραμματείας και των φορέων Π.Α. .

Διαπιστώνουμε λοιπόν πως ένα πληροφοριακό σύστημα με τις κατάλληλες αυτοματοποιήσεις θα έδινε σημαντικές λύσεις σε όλες αυτές τις διαδικασίες, ώστε να γίνουν με μεγαλύτερη ασφάλεια , ταχύτητα και όσο το δυνατόν με λιγότερη εμπλοκή του ανθρώπινου παράγοντα σε επαναλαμβανόμενες διαδικασίες.

1.2 Γιατί open-source;

Το παρόν έργο διατίθεται ως λογισμικό ανοιχτού κώδικα σε δημόσιο αποθετήριο στο GitHub, με σκοπό να δώσει τη δυνατότητα σε οποιοδήποτε άτομο να αναπτύξει περισσότερο το σύστημα κατόπιν συνεννόησης με το δημιουργό. Κάθε συνεισφορά , έχει ως σκοπό να βελτιώσει τη ποιότητα των παρεχόμενων υπηρεσιών του συστήματος και να προστεθούν νέες δυνατότητες στο σύστημα. Πιστεύουμε πως κάθε μέλος της ακαδημαϊκής κοινότητας , ή ακόμα και κάποιος εξωτερικός παράγοντας θα μπορούσε να μας βοηθήσει να αναπτύξουμε περαιτέρω αυτό το σύστημα , με σκοπό να ανταποκριθούμε στις σύγχρονες απαιτήσεις των μελών της ακαδημαϊκής μας κοινότητας.

1.3 Δομή εργασίας

Όσον αφορά τη δομή της εργασίας , αυτή διαχωρίζεται στις εξής θεματικές ενότητες/κεφάλαια:

- Κεφάλαιο 1: Πραγματεύεται ο σκοπός της και αναλύεται η υπάρχουσα κατάσταση σχετικά με τις Π.Α.

- Κεφάλαιο 2: Αναλύονται οι απαιτήσεις του συστήματος , οι τεχνολογίες υλοποίησης , οι ρόλοι και οι ενέργειες του κάθε μέλους, ο σχεδιασμός της Βάσης Δεδομένων , η διασύνδεση με τον LDAP Server του πανεπιστημίου και η αυθεντικοποίηση και οι όψεις της εφαρμογής (Views).

- Κεφάλαιο 3: Παρουσίαση των κύριων λειτουργιών της εφαρμογής και εξήγηση σεναρίων όλων των εμπλεκόμενων μελών

- Κεφάλαιο 4: Συμπεράσματα σχετικά με την υλοποίηση του συστήματος και εμβάθυνση σε μελλοντικές επεκτάσεις.

ΚΕΦ.2: Σχεδιασμός Συστήματος

2.1 Απαιτήσεις (Requirements)

Λειτουργικές Απαιτήσεις:

1. Το σύστημα απευθύνεται σε όλα τα τμήματα του Χαροκοπείου Πανεπιστημίου.
2. Το κάθε τμήμα του Χαροκοπείου Πανεπιστημίου διαθέτει δική του γραμματεία.
3. Οι φορείς απευθύνονται σε ένα ή περισσότερα τμήματα του πανεπιστημίου.
4. Οι φορείς έχουν ένα σημείο επαφής , δηλαδή έναν χρήστη που εκπροσωπεί το φορέα εντός του συστήματος.
5. Οι επιβλέποντες καθηγητές βλέπουν το αρχείο των φοιτητών που έχουν αναλάβει δική τους Π.Α.
6. Οι αιτούντες φοιτητές συμπληρώνουν 1 έως και 5 θέσεις πρακτικής άσκησης που επιθυμούν να αναλάβουν.
7. Οι αιτούντες φοιτητές μπορούν να ανεβάσουν αρχεία , όπως παρουσιολόγια , αξιολόγηση φορέα και έκθεση πεπραγμένων μετά την ολοκλήρωση της Π.Α.
8. Τα σημεία επαφής των φορέων μπορούν να βαθμολογήσουν φοιτητές που ολοκλήρωσαν τη πρακτική άσκηση στο φορέα τους.
9. Οι καθηγητές βαθμολογούν τους φοιτητές τους μετά το τέλος της πρακτικής άσκησης τους.
10. Ο φοιτητής βλέπει τη συνολική αναφορά της πρακτικής του άσκησης από τη φάση της αίτησης έως και τη ανάθεση του.
11. Η γραμματεία αποδέχεται εγγραφή νέων φορέων/φοιτητών στο σύστημα.
12. Η γραμματεία επικυρώνει/απορρίπτει αιτήσεις φορέων για αγγελίες Π.Α.
13. Τα σημεία επαφής των φορέων αποδέχονται προτάσεις πρακτικών ασκήσεων για φοιτητές.

14. Η γραμματεία προτείνει φοιτητές στους φορείς προς κάλυψη των θέσεων εργασίας.

Μη Λειτουργικές απαιτήσεις

1. Η γλώσσα του ιστότοπου είναι η αγγλική.
2. Ο μέσος χρόνος απόκρισης σε κάθε αίτημα δεν ξεπερνά τα 3 δευτερόλεπτα.
3. Οι αιτούντες έχουν πρόσβαση στα δικά τους προσωπικά στοιχεία , και αρχεία που ανεβάζουν στη σελίδα.
4. Η κάθε γραμματεία βλέπει αιτήσεις , θέσεις εργασιών , φοιτητές , φορείς και σημεία επαφής φορέων που απευθύνονται στο δικό τους τμήμα.
5. Η γραμματεία μπορεί να τροποποιήσει στοιχεία των αιτήσεων που δεν αφορούν ευαίσθητα δεδομένα των χρηστών για κανονιστικούς λόγους.
6. Η γραμματεία δε μπορεί να δημιουργήσει αιτήσεις εκ μέρους των φοιτητών.

2.2 Εργαλεία υλοποίησης (Implementation tools)

Η ανάπτυξη του ιστότοπου έγινε με τη χρήση του Django Framework , το οποίο είναι βασισμένο στη γλώσσα προγραμματισμού Python. Στη περίπτωση του Django Framework , παρατηρήσαμε πως υπάρχουν αρκετά πλεονεκτήματα σε περιπτώσεις που είναι απαραίτητο ένα περιβάλλον πολλαπλών χρηστών , με πολλαπλούς ρόλους και πολλές αρμοδιότητες.

Μας παρέχονται έτοιμες λύσεις για την αυθεντικοποίηση των χρηστών (authentication system) , με κατάλληλο περιβάλλον διαχειριστή (admin system) , εύκολη επικοινωνία/σύνδεση με τη βάση δεδομένων με τη χρήση του Django ORM , εύκολος δυναμικός σχεδιασμός της βάσης δεδομένων από τα μοντέλα (models) των εφαρμογών (Django apps) με τη χρήση των μετεγκαταστάσεων δεδομένων (migrations) , εύχρηστο template engine και αρκετά πακέτα για την υλοποίηση φορμών συμπλήρωσης στοιχείων με έμφαση στο serialization και το validation.

Σχετικά με τα γραφικά στοιχεία του UI , χρησιμοποιήθηκε η βιβλιοθήκη HTML (Hypertext Markup Language)/CSS(Cascade Style Sheets)/JavaScript Bootstrap 4.

Για να διασφαλίσουμε την cross-platform (διαλειτουργικότητα μεταξύ των ΛΣ) συμβατότητα του συστήματος , κρίνεται απαραίτητο το πακετάρισμα των εφαρμογών μας σε κοντέινερ (container).Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε το Docker τόσο για την ανάπτυξη της εφαρμογής (development) αλλά και το άνοιγμα της (deployment).

Ωστόσο για να να δώσουμε τη τελική λύση για την αρχιτεκτονική της εφαρμογής μας , χρειαστήκαμε υπηρεσίες για τη παράθεση των στατικών αρχείων και μια Β.Δ. για την αποθήκευση και διατήρηση των δεδομένων.
Για αυτό και χρησιμοποιήθηκαν τα παρακάτω εργαλεία τα οποία είναι πακεταρισμένα (containerized) για την εφαρμογή αυτή:

- Nginx για το παράθεση των στατικών αρχείων
- PostgreSQL για βάση δεδομένων.

Σε τελευταία ανάλυση , για κάποιες ιδιαίτερες ανάγκες που προέκυψαν κατά την ανάπτυξη της εφαρμογής μας χρησιμοποιήθηκαν κάποιες βιβλιοθήκες της Python και πιο συγκεκριμένα του Django Framework.
Οι σημαντικότερες εξ αυτών που αξίζουν να αναφερθούν είναι:

- Django crispy forms για την διαχείριση των Django forms.
- Gunicorn για τον HTTP server.
- Django filters για την δημιουργία φίλτρων αναζήτησης στα templates.
- Django Auth Ldap , για τη σύνδεση του authentication system της εφαρμογής με τον LDAP server του πανεπιστημίου.

2.2.1 Python

Η Python είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού υψηλού επιπέδου(high level) γενικού σκοπού. Υλοποιήθηκε και σχεδιάστηκε αρχικά από τον Guido van Rossum το 1991 και αναπτύχθηκε από την Python Software Foundation. Αναπτύχθηκε κυρίως με έμφαση στην εύκολη αναγνωσιμότητα του κώδικα και στο συντακτικό που θα επιτρέπει να εκφράζονται προγραμματιστικές ιδέες σε λιγότερες γραμμές κώδικα.

Για να επιτευχθεί η μετατροπή μιας γλώσσα υψηλού επίπεδου πρέπει να μετατραπεί σε γλώσσα μηχανής ώστε να εκτελεστεί από το εκάστοτε υπολογιστικό σύστημα ή Η/Υ.Η διαδικασία αυτή γίνεται είτε από τους διερμηνευτές(interpreters) είτε από τους μεταγλωττιστές (compilers). Στη περίπτωση της Python η διαδικασία εκτέλεσης γίνεται με τη χρήση διερμηνέα , όπου οι εντολές εκτελούνται απευθείας χωρίς να έχει προηγηθεί η μεταγλώττιση ολόκληρου προγράμματος σε γλώσσα μηχανής.

Σε υψηλό επίπεδο προγραμματισμού η Python παρέχει μια πληθώρα από δομές δεδομένων σε συνδυασμό με τη δυνατότητα της δυναμικής τυποθέτησης (dynamic typing) των μεταβλητών και της δυναμικής σύνδεσης (dynamic binding) , που την κάνουν αρκετά προσιτή για Ταχεία Ανάπτυξη Εφαρμογών (Rapid Application

Development) , όσο και για την σύνταξη κώδικα (scripts) ή τη χρήση της για εργασίες που απαιτούν ένωση μεγαλύτερων προγραμμάτων.

Η Python κατά γενική ομολογία , έχει ομαλή καμπύλη μάθησης(learning curve) για έμπειρους και αρχάριους προγραμματιστές , αυξάνει την ευαναγνωσιμότητα του κώδικα με αποτέλεσμα την εξοικονόμηση πόρων κατά την ανάπτυξη αλλά και τη συντήρηση ενός προγράμματος. Ο διερμηνέας της Python αλλά και η εκτεταμένη ενσωματωμένη βιβλιοθήκη που διαθέτει σε μορφή εκτελέσιμων αρχείων ή πηγαίου κώδικα είναι διαθέσιμα δωρεάν προς το κοινό για όλες τις γνωστές πλατφόρμες και μπορούν να διανεμηθούν ελεύθερα.

2.2.2 Django Framework

2.2.2.1 Django Framework

Το Django αποτελεί ένα web framework βασισμένο στη γλώσσα προγραμματισμού Python και ενδείκνυται για ταχείες υλοποιήσεις εφαρμογών ιστού με έμφαση στην ανάπτυξη καθαρού και πρακτικού λογισμικού κατά την ανάπτυξη τους. Αποτελεί ένα framework του οποίου η χρήση είναι δωρεάν για ιδιώτες και οργανισμούς , αλλά και ανοιχτού κώδικα στο οποίο μπορούν να συνεισφέρουν αρκετοί προγραμματιστές ανά το κόσμο και προσφέρει αρκετές λύσεις και αυτοματοποιήσεις για προγραμματιστές , με αποτέλεσμα η αφοσίωση τους να είναι στραμμένη περισσότερο στο να χτίζουν κομμάτια των εφαρμογών που είναι ζωτικά για τη κύρια λειτουργικότητα τους και όχι σε περιφερειακές και δευτερεύουσες λειτουργίες που αφορούν π.χ. την αυθεντικοποίηση των χρηστών ή την ανάπτυξη πάνελ διαχείρισης χρηστών(admin user dashboard).

Το Django αναπτύχθηκε αρχικά μεταξύ του 2003 και 2005 από μια ομάδα στο διαδίκτυο οι οποίοι ήταν υπεύθυνοι για να ανταπτύσσουν και να συντηρούν ειδησεογραφικούς ιστότοπους.

Έπειτα από την ανάπτυξη αρκετών ιστότοπων , η ομάδα ξεκίνησε να υλοποιεί και να επαναχρησιμοποιεί αρκετό κώδικα και μοτίβα σχεδίασης.

Ο κοινός κώδικας που χρησιμοποιήθηκε εξελίχθηκε σε ένα γενικό framework ανάπτυξης εφαρμογών ιστού , και έγινε λογισμικό ανοιχτού κώδικα ως έργο “Django” τον Ιούλιο του 2005.[3]

Το Django συνεχίζει να μεγαλώνει και αν εξελίσσεται , από τη πρώτη του έκδοση 1.0 το Σεπτέμβριο του 2008 έως τη πιο πρόσφατη έκδοση 4.0 που αναμένεται να κυκλοφορήσει εντός του 2022.Σε κάθε μια από τις νεες εκδόσεις προστίθενται νέες λειτουργίες και διορθώνονται σφάλματα στο κώδικα.Οι προστιθέμενες λειτουργίες μπορούν να προσφέρουν υποστήριξη σε νέους τύπους ΒΔ , template engines ,

caching και τη προσθήκη νέων γενικών όψεων(Views) είτε σε μορφή συναρτήσεων είτε κλάσεων(οι οποίες μειώνουν σημαντικά τη ποσότητα του κώδικα που πρέπει να αναπτύξουν οι προγραμματιστές για ένα συγκεκριμένο αριθμό προγραμματιστικών εργασιών).[4]

Η αρχιτεκτονική στην οποία βασίζεται το Django είναι η MVT(Model-View-Template).Το MVT είναι ένα μοτίβο σχεδίασης λογισμικού που συναντάται κατά την ανάπτυξη εφαρμογών ιστού.[5]

Η δομή του MVT αποτελείται από τα εξής τρία κομμάτια:

Model: Πρόκειται για το μοντέλα που διαδραματίζουν το ρόλο της διεπαφής για τα δεδομένα μιας εφαρμογής. Η λογική δομή των δεδομένων πίσω από μια εφαρμογή η οποία αναπαρίσταται σε μια Β.Δ. (Σχεσιακές Βάσεις Δεδομένων όπως MySql , Postgres).

Παράδειγμα ενός μοντέλου και η αντιστοίχιση του σε μια σχεσιακή βάση δεδομένων:

```
You, 4 seconds ago | 1 author (You)
class Address(models.Model):
    country = models.CharField(max_length=30, validators=[alphabetic])
    city = models.CharField(max_length=40, validators=[alphabetic])
    street_name = models.CharField(max_length=100, validators=[alphabetic])
    street_number = models.IntegerField(
        validators=[MinValueValidator(0), MaxValueValidator(9999)]
    )
    postal_code = models.IntegerField(
        validators=[MinValueValidator(0), MaxValueValidator(99999)]
    )

    def __str__(self):
        return (
            self.country
            + ", "
            + self.city
            + ", "
            + self.street_name
            + ", "
            + str(self.street_number)
        )
```

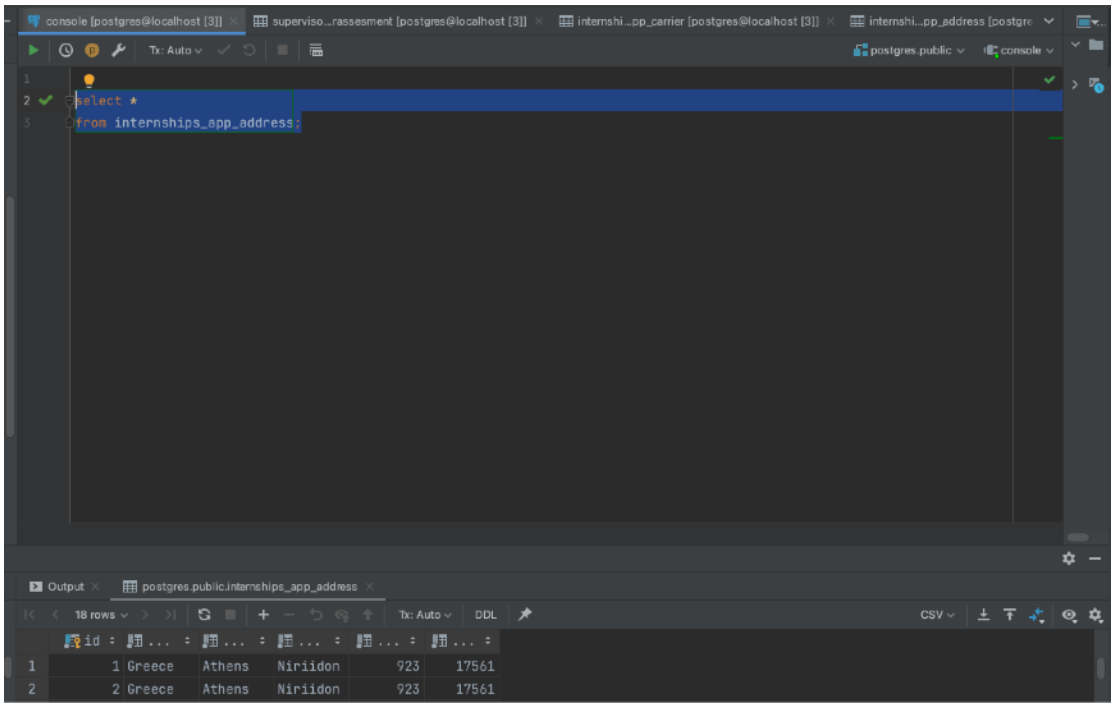
Μοντέλο Django 1

Παραπάνω βλέπουμε στο αρχείο models.py κάτω από τον φάκελο internships_app ένα μοντέλο το οποίο αναπαριστά την Διεύθυνση Κατοικίας ενός χρήστη.

Η αντιστοίχιση του μοντέλου σε ένα σύστημα ΣΧΒΔ έχει αυτή τη μορφή:

(όνομα εφαρμογής)_(όνομα μοντέλου)

Κατά συνέπεια όταν είμαστε στο περιβάλλον της Β.Δ. , η οντότητα αυτή καλείται με το όνομα internshiops_app_address:



Εγγραφή μοντέλου Django στην ΒΔ

Στο παραπάνω screenshot εκτελούμε ένα ερώτημα PostgreSQL με τη χρήση του προγράμματος DataGrip της Jetbrains με σκοπό να αντλήσουμε διευθύνσεις κατοικίας που έχουν δημιουργηθεί από την εφαρμογή.

View: Πρόκειται για τις όψεις(Views) τα οποία αποτελούν διεπαφές χρήστη , πιο συγκεκριμένα το σύνολο των γραφικών στοιχείων όταν φορτώνεται μια σελίδα στο πρόγραμμα περιήγησης σας. Η αναπαράσταση γίνεται μέσω HTML/CSS/JavaScript και Jinja αρχείων.

Στο Django μπορούμε να διακρίνουμε δύο τύπους όψεων , τις function-based όψεις και τις class-based όψεις.

Τα Function Based Views συγγράφονται με τη χρήση συναρτήσεων της Python οι οποίες λαμβάνουν ως όρισμα ένα αντικείμενο τύπου HttpRequest και επιστρέφουν ένα αντικείμενο τύπου HttpResponse. Τα Function Based Views γενικά χωρίζονται με βάση 4 βασικές στρατηγικές , που βασίζονται σε υλοποιήσεις τύπου CRUD (Create , Retrieve , Update , Delete) ή αλλιώς (Δημιουργία , Ανάκτηση , Ενημέρωση , Διαγραφή).Το CRUD είναι η δομικό στοιχείο για όλα τα frameworks που χρησιμοποιούνται για ανάπτυξη εφαρμογών ιστού.

Τα Class-based Views μας παρέχουν έναν εναλλακτικό τρόπο να υλοποιούμε τις όψεις δίχως την χρήση των συναρτήσεων. Δεν αντικαθιστούν τις function-based όψεις , αλλά έχουν σημαντικές διαφορές και πλεονεκτήματα όταν συγκρίνονται με τα function-based views:

- Οργάνωση του κώδικα πάνω σε συγκεκριμένες μεθόδους του HTTP(GET , POST , PUT , PATCH , DELETE , UPDATE , κτλπ) και μπορούμε για κάθε μια HTTP μέθοδο να υλοποιήσουμε μια συνάρτηση εντός της κλάσης του Class based view , αποφεύγοντας τις διακλαδώσεις κατά συνθήκη κατά την εγγραφή του View.
- Τεχνικές αντικειμενοστραφούς προγραμματισμού όπως τα mixins(χρήση πολλαπλής κληρονομικότητας) η οποία μπορεί να χρησιμοποιηθεί με σκοπό να υλοποιηθούν επαναχρησιμοποιούμενα τμήματα κώδικα(reusable code components).

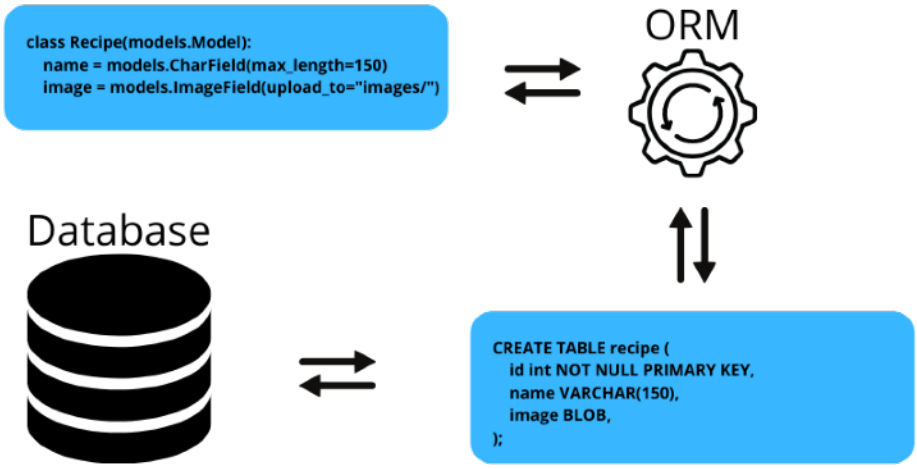
Τα Class-based Views είναι απλούστερα και πιο αποτελεσματικά κατά τη διαχείριση τους συγκριτικά με τα Function Based Views.Ένα Function-based View με χιλιάδες γραμμές κώδικα μπορεί να μετατραπεί σε ένα Class-based View με κάποιες εκατοντάδες γραμμές κώδικα. Σε αυτό συνεισφέρει η χρήση των τεχνικών του Αντικειμενοστραφούς προγραμματισμού που έχει ως ένα από τους βασικούς στόχους του την επαναχρησιμοποίηση του κώδικα.[5]

Template: Ένα template εμπεριέχει στατικά αρχεία των επιθυμητών HTML αρχείων που λαμβάνεται ως έξοδος από την εφαρμογή , όπως επίσης και μια ιδιάζουσα σύνταξη η οποία περιγράφει τους τρόπους με τους οποίους γίνεται η εισαγωγή δυναμικού περιεχομένου στη σελίδα.

2.2.2.2 Django ORM

Το Django παρέχει έναν ενσωματωμένο μηχανισμό Αντιστοίχισης Αντικειμένων(ORM), ο οποίος μας επιτρέπει να χρησιμοποιούμε σχεσιακές βάσεις δεδομένων με τις αντίστοιχες κλάσεις της Python προσφέροντας μας ένα επίπεδο αφαίρεσης(level of abstraction).

Το Django ORM επιτρέπει στους προγραμματιστές να αλληλοεπιδρούν με τη ΒΔ , χωρίς να χρησιμοποιούν γλώσσες προγραμματισμού Σχεσιακών Βάσεων Δεδομένων αλλά αντικειμενοστραφή κώδικα Python. Ο μηχανισμός Αντιστοίχισης Αντικειμένων μετατρέπει αυτόματα τον κώδικα της Python σε SQL (Structured Query Language) , σε μια γλώσσα όπου ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων μπορεί να καταλάβει. Στο πυρήνα του μηχανισμού αυτού , έχουμε ένα επίπεδο αφαίρεσης για την αλληλεπίδραση της Django εφαρμογής μας με τη ΒΔ και αντίστροφα. [6]



Λειτουργία ORM

Διαγραμματική εξήγηση της μετατροπής του κώδικα python ενός μοντέλου σε μια εντολή SQL που δημιουργεί ένα νέο πίνακα με όνομα "recipe" με τα αντίστοιχα πεδία που έχουν δηλωθεί στη κλάση "Recipe".

2.2.2.3 Django Crispy Forms

Το πακέτο Django-crispy-forms είναι μια βιβλιοθήκη/εφαρμογή , η οποία χρησιμοποιείται για τη διαχείριση των φορμών του Django.Μέσω του crispy-forms , μπορούμε να προσαρμόσουμε τις ιδιότητες μιας φόρμας από τη πλευρά του εξυπηρετητή(server/backend) δίχως την ανάγκη να παραμετροποιούμε τον κώδικα στα templates.Το πακέτο αυτό μας βοηθά μας βοηθά να εμπλουτίσουμε τη λειτουργικότητα των φορμών του Django , τα templates είναι αποδοτικότερα κατά τη χρήση τους και μπορούμε να αναπτύξουμε σύνθετα σενάρια χρήσης στο γραφικό μας περιβάλλον με λιγότερο κώδικα.Τέλος , με τη χρήση αυτού του πακέτου μπορούμε να προβούμε σε αλλαγές στην εμφάνιση και την αίσθηση(look and feel) με τη χρήση τάξεων και κλάσεων της CSS για όλη τη φόρμα και σε μεμονωμένα σημεία π.χ πεδία , ετικέτες πεδίων και κουμπιά.[7]

2.2.3 Bootstrap 4

Το Bootstrap είναι ένα μια ανοιχτού κώδικα εργαλειοθήκη (toolkit) για την ανάπτυξη κώδικα HTML , CSS και JavaScript.

Το Bootstrap είναι το πιο ευρέως χρησιμοποιούμενο πλαίσιο(framework) σε ολόκληρο τον κόσμο.Ξεκίνησε το 2011 από τους Mark Otto και Jacob Thornton.

- Το Bootstrap μας παρέχει μια **ποικιλία κλάσεων CSS** - CSS classes που μας βοηθούν στην υλοποίηση του σχεδιασμού του ιστοτόπου μας.
- Επίσης, προσφέρει **έτοιμα συστατικά** - ready-made web components (π.χ. καρουσέλ - carousel, γραμμή πλοήγησης - navigation bar, ακορντεόν - accordion) που μας βοηθούν να δημιουργήσουμε λειτουργικούς, διαδραστικούς και σύγχρονους ιστότοπους.
- Το Bootstrap υποστηρίζει επίσης "**συνεπή τυπογραφία** - consistent typography", η οποία είναι η συνεπής δομή του κειμένου στην ιστοσελίδα.
- Φυσικά, εκτός από τα παραπάνω, το bootstrap έχει σχεδιαστεί για να παρέχει **responsive, mobile-first websites.**[8]

2.2.4 PostgreSQL

Η PostgreSQL είναι μια ισχυρή , ανοιχτού κώδικα αντικειμενοστραφής σχεσιακή βάση δεδομένων που επεκτείνει τις δυνατότητες της γλώσσας SQL σε συνδυασμό με πολλές λειτουργίες που βοηθούν σε ασφαλή αποθήκευση και στη κλιμάκωση των πιο σύνθετων φορτίων εργασιών.Οι ρίζες της PostgreSQL μας φέρνουν στο έτος 1986 , όταν το έργο POSTGRES ξεκίνησε στο Πανεπιστήμιο του Μπέρκλεϊ της Καλιφόρνια και έχει πάνω από 30 χρόνια ανάπτυξης στη βασική του πλατφόρμα.

Η PostgreSQL έχει αποκτήσει ισχυρή φήμη για την αποδεδειγμένη αρχιτεκτονική της , την αξιοπιστία , την ακεραιότητα των δεδομένων , το στιβαρό σύνολο λειτουργιών , την επεκτασιμότητα και την αφοσίωση της κοινότητας ανοικτού κώδικα πίσω από τη συνεχή ανάπτυξη και παράδοση λογισμικού με σκοπό τη διάθεση αποδοτικών και καινοτόμων λύσεων στον τομέα των λογισμικών. Η PostgreSQL εκτελείται σε όλες τις μεγάλες κοινότοπες πλατφόρμες και λειτουργικά συστήματα , έχει συμμορφωθεί με το πρότυπο ACID από το 2001 , και έχει ισχυρές επεκτάσεις(add-ons), όπως η ευρέως γνωστή PostGIS για διαχείριση γεωχωρικών δεδομένων. Δεν αποτελεί έκπληξη το γεγονός πως η PostgreSQL αποτελεί τη βασικότερη επιλογή σχεσιακής βάσης δεδομένων για πολλούς προγραμματιστές και οργανισμούς.

Η PostgreSQL έχει αρκετά χαρακτηριστικά τα οποία στοχεύουν στο να βοηθήσουν τους προγραμματιστές να χτίσουν εφαρμογές , διαχειριστές συστημάτων να διασφαλίσουν την ακεραιότητα των δεδομένων τους και να φτιάξουν περιβάλλοντα

τα οποία είναι ανεκτικά σε σφάλματα και βοηθούν στη διαχείριση των δεδομένων ανεξάρτητα από το συνολικό τους μέγεθος. Επιπροσθέτως , η PostgreSQL πέρα από δωρεάν με διαθέσιμο πηγαίο κώδικα είναι σε μεγάλο βαθμό επεκτάσιμη. Δίνονται αρκετές δυνατότητες σε προγραμματιστές να δημιουργήσουν δικούς τους τύπους δεδομένων , δικές τους συναρτήσεις αλλά ακόμα και να γράψουν κομμάτια κώδικα σε διαφορετικές γλώσσες προγραμματισμού δίχως να απαιτείται η μεταγλώττιση της ΒΔ.

Η PostgreSQL προσπαθεί να συμμορφωθεί με το πρότυπο της SQL , με προϋπόθεση πως η προσπάθεια αυτή δεν έρχεται σε σύγκρουση με τις παραδοσιακές λειτουργίες ή να γίνουν μη επιθυμητές αλλαγές στην αρχιτεκτονική της. Πολλές από τις λειτουργίες που απαιτούνται από το πρότυπο της SQL υποστηρίζονται , ωστόσο μερικές φορές με ελαφρώς διαφορετικό συντακτικό ή τρόπους λειτουργίας. Περαιτέρω ενέργειες προς την συμμόρφωση με το πρότυπο είναι αναμενόμενες οποιαδήποτε στιγμή στο μέλλον. Στην έκδοση 14 που κυκλοφόρησε τον Σεπτέμβριο του 2021 , η PostgreSQL συμμορφώνεται με 170 από τις 179 υποχρεωτικές λειτουργίες της SQL:2016.Κατά το χρονικό διάστημα που γίνεται αναφορά σε αυτό , καμία Σ.Χ.Β.Δ. δεν έχει συμμορφωθεί πλήρως με αυτό το πρότυπο.

Παρακάτω θα εντοπίσετε μια σειρά από λειτουργίες και δυνατότητες που βρίσκονται στην PostgreSQL:

- ANSI SQL 89 , 92 , 93
- 100% συμβατή με ACID και πλήρη υποστήριξη commit και rollback.
- Online αντίγραφα ασφαλείας: υψηλότερη ασφάλεια και διαθεσιμότητα των δεδομένων.
- Τύποι δεδομένων: numeric, decimal, smallint , integer, bigint , real, double , serial, char, varchar, bit, text, date, time, timestamp, interval, boolean, network address, geometric types και πολλά άλλα.
- Δυνατότητα δημιουργίας νέων τύπων δεδομένων από τους χρήστες.
- Αποθήκευση BLOBS (binary large objects), συμπεριλαμβανομένων αρχείων κειμένου, ήχου, εικόνων ή βίντεο.
- Πλήρης υποστήριξη συναρτήσεων συγκεντρωτικών αποτελεσμάτων (GROUP_BY) όπως COUNT, SUM, AVG, MIN, MAX, STDDEV and VARIANCE. Δυνατότητα δημιουργίας νέων συγκεντρωτικών συναρτήσεων εφόσον χρειαστεί.
- Υποστήριξη όλων των τύπων ενώσεων (cross , inner , outer , left , right , full , natural).
- Συναρτήσεις ορισμένες από τον χρήστη , οι οποίες μπορούν να γραφούν σε πολλές γλώσσες προγραμματισμού όπως C, SQL , PL/pgSQL , TCL , Perl , Python and Ruby.
- Περιβάλλον ανάπτυξης γλωσσών προγραμματισμού όπως Perl , Python, Zope, PHP, TCL/TK, ODBC, JDBC, C/C++, Embedded SQL, Delphi/Kylix/Pascal, VB , ASP, Java.
- Βιβλιοθήκη συναρτήσεων και τελεστών με ορισμένες προεγκατεστημένες συναρτήσεις όπως math, date/time, string, geometric, formatting κ.α.

- Συναρτήσεις trigger μπορούν να οριστούν από οποιαδήποτε γλώσσα προγραμματισμού που υποστηρίζει server όπως C ή PL/pgQL.
- Προσωρινοί πίνακες οι οποίοι σβήνονται αυτόματα μετά το τέλος της συνόδου.
- Μοντέλο ασφαλείας ομάδας/χρήστη. Η πρόσβαση στο διακομιστή της βάσης δεδομένων μπορεί να περιορίζεται από το χρήστη , κεντρικό υπολογιστή ή μια βάση δεδομένων.
- Το μέγεθος του πίνακα και της βάσης δεδομένων είναι σχεδόν απεριόριστο. Απεριόριστες καταχωρήσεις και ευρετήρια ανα πίνακα

Υπάρχουν πολλές περισσότερες λειτουργίες που προσφέρει η PostgreSQL , αλλά για τους σκοπούς αυτής της εργασίας δε θεωρούμε πως αυτή η εμβάθυνση εξυπηρετεί το στόχο της. Επιπροσθέτως , η PostgreSQL είναι σε μεγάλο βαθμό επεκτάσιμη: πολλές από τις λειτουργίες της , όπως τα ευρετήρια έχουν ορισμένες προγραμματιστικές διεπαφές εφαρμογών(Application Programming Interfaces) οι οποίες επιδέχονται παραμετροποιήσεις για την επίλυση συγκεκριμένων προβλημάτων που αντιμετωπίζει ο κάθε χρήστης της.

Ένα από τα μεγαλύτερα πλεονεκτήματα που είναι άξιο αναφοράς είναι το υψηλό επίπεδο κλιμάκωσης όταν υπάρχει η απαίτηση για διαχείριση μεγάλης ποσότητας δεδομένων με πολλούς παράλληλους χρήστες που δύναται να εξυπηρετήσει ταυτόχρονα. Υπάρχουν ενεργές συστάδες(clusters) υπολογιστών σε παραγωγικές υποδομές οι οποίες μπορούν να διαχειριστούν πολλά terabytes έως και petabytes δεδομένων.[9] .

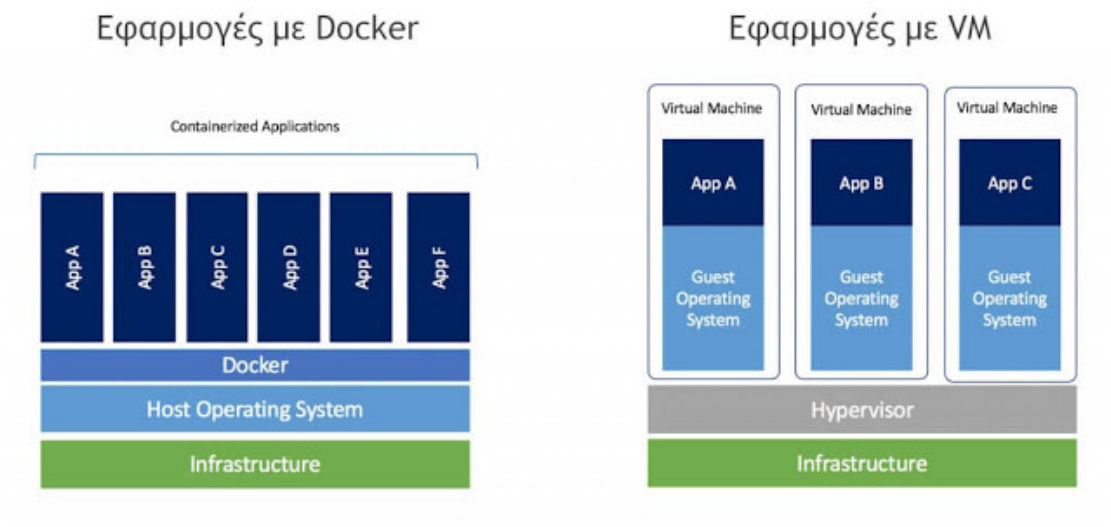
2.2.5 Nginx

Το nginx είναι ένα λογισμικό ανοιχτού κώδικα για διακομιστές ιστού , με αντίστροφους διακομιστές μεσολάβησης , μέσα προσωρινής αποθήκευσης δεδομένων , εργαλεία εξισορρόπησης φορτίων και ροή πολυμέσων – και χρησιμοποιείται ευρέως για την ελαφριά αρχιτεκτονική υψηλής απόδοσης
Ο διακομιστής nginx είναι ένας δωρεάν διακομιστής ιστού ανοικτού κώδικα. Λειτουργεί σε λειτουργικά συστήματα Linux/Unix 64 bit και χρησιμοποιείται ευρέως για ιστότοπους υψηλής απόδοσης λόγω της ελαφριάς αρχιτεκτονικής του σε σύγκριση με τους διακομιστές Apache.
Το Nginx είναι ένας διακομιστής web υψηλής απόδοσης με ελαφριά αρχιτεκτονική. Λειτουργεί σε λειτουργικά συστήματα Linux/Unix 64 bit και χρησιμοποιείται ευρέως για ιστότοπους υψηλής επισκεψιμότητας επειδή μπορεί να χειριστεί περισσότερες ταυτόχρονες συνδέσεις από τους διακομιστές Apache. Το Nginx έχει αποδειχθεί ότι ξεπερνά τον Apache στα σημεία αναφοράς , ειδικά κατά την προβολή στατικών αρχείων.
Το αρχείο διαμόρφωσης Nginx βρίσκεται στο /etc/Nginx/Nginx.conf και περιέχει οδηγίες που ελέγχουν τη λειτουργία του διακομιστή. Οι μονάδες Nginx βρίσκονται στον κατάλογο /usr/lib64/Nginx/modules. Οι πιο συχνά χρησιμοποιούμενες ενότητες είναι:

πυρήνας – βασική λειτουργικότητα Nginx
SSL – παρέχει υποστήριξη SSL
διακομιστής μεσολάβησης – ενεργοποιεί την αντίστροφη υποστήριξη διακομιστή μεσολάβησης
geo – ενεργοποιεί την υποστήριξη γεωγραφικής τοποθεσίας
συμβάν – ενεργοποιεί το μοντέλο προγραμματισμού που βασίζεται σε συμβάντα
Υπάρχουν πολλές άλλες διαθέσιμες ενότητες και η λίστα αλλάζει συχνά. Μπορείτε να βρείτε μια τρέχουσα λίστα λειτουργικών μονάδων Nginx στο wiki Nginx.[10]

2.2.6 Docker

Το Docker είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα, η οποία μας επιτρέπει να εφαρμόσουμε εικονικοποίηση(virtualization) σε επίπεδο λειτουργικού συστήματος.Προσφέρεται η δυνατότητα εγκατάστασης εφαρμογών και υπηρεσιών της επιθυμίας του εκάστοτε χρήστη δίχως εγκατάσταση επιπρόσθετου λειτουργικού συστήματος, σε απομονωμένο περιβάλλον αξιοποιώντας πόρους οι οποίοι βρίσκονται στο φυσικό μηχάνημα(host machine) όπως Κεντρική Μονάδα Επεξεργασίας, Κάρτα Γραφικών και Μνήμη Τυχαίας Προσπέλασης με οικονομικό τρόπο.
Οι εφαρμογές αυτές τρέχουν σε μία “φούσκα” (container), τα περιεχόμενα της οποίας είναι απομονωμένα από το υπόλοιπο περιβάλλον του υπολογιστή και μας δίνεται η δυνατότητα να τη… “σκάσουμε” όποτε θέλουμε, για να διαγράψουμε τα πάντα από το σύστημα μας. [11]



Αρχιτεκτονική Docker

2.3 Ρόλοι του συστήματος

Όπως είχε αναφερθεί και προηγουμένως στην ενότητα 1.1 , οι ρόλοι του συστήματος είναι τέσσερις και είναι οι εξής:

1. **Προπτυχιακός Φοιτητής (Undergraduate Student)**

2. **Σημείο επαφής Φορέα (Carrier Node)**

3. **Επιβλέπων Καθηγητής (Supervisor)**

4. **Υπάλληλος Γραμματείας (Secretarian)**

5. **Διαχειριστής Εφαρμογής (Application admin)**

2.3.1 Φοιτητής

Οι φοιτητές βρίσκονται στο προπτυχιακό επίπεδο από όλα τα τμήματα του πανεπιστημίου(Διαιτολογία , Πληροφορική , Γεωγραφία , Οικιακή Οικονομία) και έχουν κοινές όψεις(Views) κάτω από την εφαρμογή(Django app) **applicant**(αιτών). Τους δίνεται η δυνατότητα να συμπληρώσουν την αίτηση με τις θέσεις πρακτικής άσκησης των προτιμήσεων τους , με σειρά προτεραιότητας από το 1(ένα) έως και το 5(πέντε) για όσο είναι ανοιχτή η περίοδος δήλωσης ενδιαφέροντος για τις πρακτικές ασκήσεις.Επίσης μπορούν να ενημερώνουν την αίτηση τους , αλλάζοντας είτε τις θέσεις , είτε τη προτεραιότητα των θέσεων για όσο είναι ανοιχτή η περίοδος δήλωσης ενδιαφέροντος για τις πρακτικές ασκήσεις(**Application Period**).

Όταν οριστικοποιούνται οι τελικές αναθέσεις των πρακτικών ασκήσεων , οι φοιτητές βλέπουν αναλυτικά στη καρτέλα εξέλιξης της αίτησης τους το φορέα εισαγωγής τους και τη θέση εργασίας για την οποία έχουν επιλεχθεί.

Μετά το πέρας των πρακτικών ασκήσεων οι φοιτητές μπορούν να ανεβάσουν το τελικό αρχείο της πρακτικής το οποίο περιλαμβάνει τα εξής 3(τρία) δικαιολογητικά:

1. Ερωτηματολόγιο αξιολόγησης φορέα. (αρχείο .pdf <2 MB)
2. Παρουσιολόγιο (αρχείο .pdf <2 MB)
3. Έκθεση Πεπραγμένων Π.Α. (αρχείο .pdf <2 MB)

Τεχνικά , για κάθε τμήμα ορίζεται μια περίοδος υποβολής αιτήσεων δικαιολογητικών περάτωσης πρακτικής άσκησης (Internship Report Period) κατά την οποία οι φοιτητές μπορούν να οδηγηθούν στο σημείο της λεπτομερούς αναφοράς της εξέλιξης της αίτησης της πρακτικής άσκησης και κάτω από το σημείο που αναγράφεται η ανάθεση τους να πατήσουν το κουμπί "Create Internship Report". Όσο είναι ενεργό το προαναφερθέν διάστημα, οι φοιτητές μπορούν να τροποποιούν και να ενημερώνουν την αίτηση τους, μέχρις ότου ληφθεί η έγκριση της από τη γραμματεία του τμήματός τους.

2.3.2 Σημείο επαφής Φορέα

Το Σ.Ε.Φ.(Σημείο Επαφής Φορέα) ορίζει έναν υπάλληλο του εκάστοτε φορέα που συμμετέχει στο πρόγραμμα Π.Α. του πανεπιστημίου.Η αρμοδιότητα του πριν γίνει μέλος στο σύστημα είναι να συμπληρώνει τα στοιχεία του φορέα που εκπροσωπεί , καθώς και τα προσωπικά του στοιχεία. Τα Σ.Ε.Φ. μοιράζονται τα Views κάτω από την εφαρμογή carrier(φορέας).Εντός της συγκεκριμένης εφαρμογής(Django app) έχουμε όψεις(Views) που δίνουν τη δυνατότητα σε ένα Σ.Ε.Φ. να δημιουργεί θέσεις εργασίας με μοναδικό αναγνωριστικό/κωδικό εργασίας , να τροποποιεί τα στοιχεία της και να τις διαγράφει. Τα παραπάνω δύναται να λάβουν χώρα , κατά το διάστημα ενδιαφέροντος συμπλήρωσης εργασιών(**Carrier Assignment Period**) και για όσο η γραμματεία του αντίστοιχου τμήματος που η θέσης εργασίας απευθύνεται δεν έχει επικυρώσει ή ακυρώσει αυτή τη θέση.

Κατά τη περίοδο ορισμού αναθέσεων εργασιών(**Application**) , όπου απονέμονται οι θέσεις εργασίας σε αιτούντες φοιτητές. Τα Σ.Ε.Φ. λαμβάνουν κάποιες προτάσεις για αναθέσεις εργασιών από τη γραμματεία του αρμόδιου τις οποίες βλέπουν, και με βάση το προφίλ του υποψηφίου αποφασίζουν αν θα οριστικοποιήσουν ή απορρίψουν την ανάθεση του. Τα στοιχεία επικοινωνίας του ενδιαφερόμενου φοιτητή κοινοποιούνται στους φορείς και μπορούν οι ίδιοι φορείς να έρθουν σε επικοινωνία εκτός τη πλατφόρμας με το φοιτητή , ώστε να διεξάγουν μια συνέντευξη από κοινού.

Μετά το πέρας της Π.Α. τα Σ.Ε.Φ. είναι υπεύθυνα να βαθμολογήσουν τους ασκούμενους φοιτητές/τριες που έκαναν τη Π.Α. στο φορέα τους συμπληρώνοντας κάποια σχόλια, τη βαθμολογία του 0-10, και υποβάλλοντας τα παρακάτω δικαιολογητικά:

- Έκθεση αξιολόγησης φοιτητή (αρχείο .pdf < 2MB)
- Έκθεση αξιολόγησης πεπραγμένων φοιτητή(αρχείο . pdf <2MB)

2.3.3. Επιβλέπων Καθηγητής

Ο Επιβλέπων Καθηγητής ορίζεται ένας καθηγητής που έχει υπό την επίβλεψη του ορισμένες Π.Α. για θέσεις εργασίας που αφορούν το τμήμα του.

Οι όψεις(Views) που μοιράζονται οι επιβλέποντες καθηγητές βρίσκονται κάτω από την εφαρμογή **supervisor.** Σε αυτές τις όψεις , οι καθηγητές μπορούν να δουν τους φοιτητές που έχουν αναλάβει τις εργασίες τους για τη Π.Α. Επίσης , σε αυτή την εφαρμογή προσφέρεται και η δυνατότητα στους καθηγητές να βαθμολογήσουν τους φοιτητές για μια ανάθεση, εφόσον έχει προηγηθεί η αξιολόγηση του φορέα και υποβολή των κατάλληλων δικαιολογητικών από τους φοιτητές. Στη συγκεκριμένη περίπτωση βλέπουν μια αναλυτική αναφορά με την εξέλιξη της Π.Α άσκησης του φοιτητή , με τα υποβληθέντα δικαιολογητικά από τους προαναφερθέντες. Κατά την αξιολόγηση του φοιτητή, ο καθηγητής συμπληρώνει κάποια σχόλια, τη τελική βαθμολογία του φοιτητή καθώς και το εξής δικαιολογητικό:

- Έκθεση αξιολόγησης φοιτητή (αρχείο .pdf < 2MB)

2.3.4 Υπάλληλος Γραμματείας

Ως υπάλληλος γραμματείας κατά το παρόν σύστημα , ορίζεται ως **ένας** και **μοναδικός** χρήστης που έχει την αρμοδιότητα να ελέγχει τις αιτήσεις των Π.Α. και τις εγγραφές την ενεργοποίηση των αιτούντων φοιτητών και φορέων για το τμήμα του.Τα τμήματα του πανεπιστημίου είναι 4 (τέσσερα) και είναι τα εξής:

1. Πληροφορικής και Τηλεματικής
2. Διατροφής και Διαιτολογίας
3. Γεωγραφίας
4. Βιώσιμης Οικονομίας και Ανάπτυξης

Οι όψεις(Views) των υπαλλήλων της γραμματείας είναι κοινές και βρίσκονται κάτω από την εφαρμογή **secretary.** Οι υπάλληλοι των γραμματειών κάθε τμήματος έχουν τη δυνατότητα να εγκρίνουν και να απορρίπτουν φοιτητές και φορείς στο σύστημα για το τμήμα τους.
Επίσης , κατά το χρονικό διάστημα ενδιαφέροντος συμπλήρωσης εργασιών (**Carrier Assignment Period**) εγκρίνουν θέσεις εργασίας που αναρτούν οι φορείς με στόχο να δημιουργηθεί η τελική λίστα αγγελιών εργασιών για το δικό τους τμήμα.

Οι υπάλληλοι της γραμματείας έχουν τη δυνατότητα κατά τη περίοδο της ανάθεσης εργασιών (**Assignment Period)** να δημιουργούν προτάσεις για θέσεις εργασίας με συγκεκριμένους φοιτητές που θα καλύψουν τις θέσεις αυτές, οι οποίες κοινοποιούνται στους αντίστοιχους φορείς που συναινούν (**Consent**) ή όχι σε κάθε υποψήφια ανάθεση.

Τέλος, οι γραμματείες κάθε τμήματος ελέγχουν τις αιτήσεις για την αξιολόγηση των φοιτητών και τα επισυναπτόμενα δικαιολογητικά, εφόσον δεν υπάρχει κάποια εκκρεμότητα ή παράλειψη τότε οριστικοποιούν το αρχείο. Στη περίπτωση που οριστηκοποιηθεί το αρχείο υποβολής του φοιτητή για την αξιολόγηση του , τότε οι φορείς έχουν τη δυνατότητα να αξιολογήσουν τους φοιτητές με βάση την έκθεση πεπραγμένων και στη συνέχεια οι καθηγητές για δικούς τους ασκούμενους φοιτητές.

2.3.5 Admin

Admin , ή αλλιώς διαχειριστής της εφαρμογής έχει πρόσβαση σε όλα τα δεδομένα του συστήματος. Στη δική του περίπτωση δεν έχουν αναπτυχθεί ιδιαίτερα views, αλλά έχουμε τροποποιήσει κυρίως τις ρυθμίσεις (configurations) του Django Framework και μέσω διαφόρων οδηγιών εξασφαλίζουμε τη παρουσίαση συγκεκριμένων οντοτήτων στη Βάση Δεδομένων στο δικό του προφίλ στο σύστημα.Σε κάθε εφαρμογή του Django διατίθεται ένα αρχείο **admin.py** στο οποίο εισάγουμε τα μοντέλα(models) τα οποία χρειαζόμαστε στο περιβάλλον διαχειριστή και μπορούμε να δώσουμε κάποιες οδηγίες για τον τρόπο με τον οποίο θα προβάλλονται μέσω της χρήσης των φορμών του Django (forms).

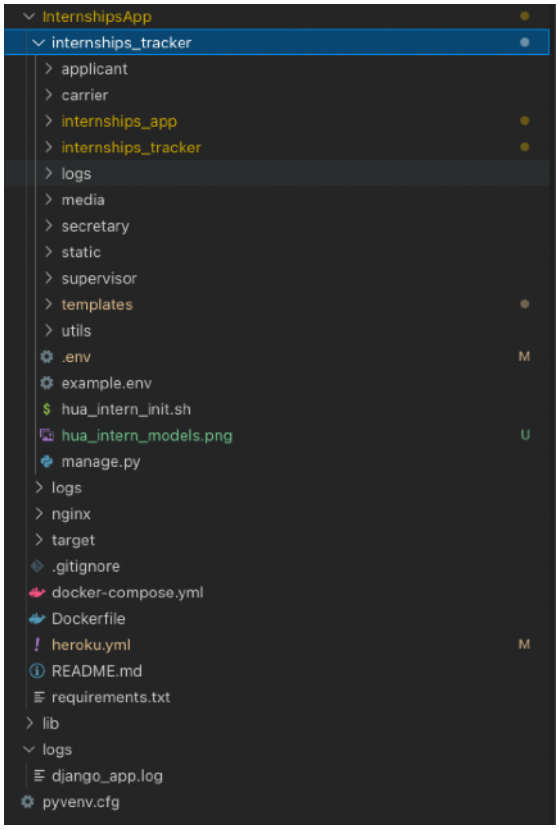
Επίσης , μέσω του διαχειριστικού περιβάλλοντος έχουμε πρόσβαση σε όλο το user base (σύνολο χρηστών) της εφαρμογής με δικαιώματα τροποποίησης όλων των πεδίων τους. Μέσω του περιβάλλοντος διαχειριστή μπορούμε να φτιάξουμε δοκιμαστικούς χρήστες οποιουδήποτε ρόλου , ή ακόμα και σε πραγματικό σενάριο τους υπαλλήλους της κάθε γραμματείας.

Ο admin μπορεί να ορίσει τα διαστήματα της εξέλιξης της Π.Α., από το Carrier Assignment Period μέχρι και το Internship Report Period.

Στο admin περιβάλλον συνιστούμε πως διαθέτουν πρόσβαση άτομα τα οποία είναι εξοικειωμένα με το τρόπο λειτουργίας της εφαρμογής , διότι ενδεχομένως οι αλλαγές σε δεδομένα να επιφέρουν και αλλαγές στη συμπεριφορά του συστήματος.

2.4 Δομή του project

Η βασική δομή της εφαρμογής μας αποτελείται από 3 βασικά στοιχεία τη Django εφαρμογή, τις ρυθμίσεις για πακετάρισμα των εφαρμογών και το σύστημα αρχείων για το διακομιστή nginx.



Σύστημα αρχείων εφαρμογής

Στο παραπάνω στιγμιότυπο παρατηρούμε τη δομή του συστήματος αρχείων της

εφαρμογής μας. Ο φάκελος "InternshipsApp" περικλείει το σύνολο της containerized εφαρμογής μας , με τα αντίστοιχα συστήματα αρχείων και τις ρυθμίσεις που απαιτούνται για την εφαρμογή Django του συστήματος μας.

Ο φάκελος "nginx" για τον διακομιστή Nginx. και κάποια αρχεία που χρησιμεύουν στην δημιουργία των docker containers όπως **docker-compose.yml** και **Dockerfile**. Επίσης ο φάκελος logs , χρησιμεύει για την αποθήκευση των logs(αρχείο δραστηριοτήτων) της εφαρμογής αλλά και του container της Django εφαρμογής.

Ο εμφωλευμένος φάκελος "internships_tracker" αποτελεί τον φάκελο του Django project μας , το οποίο διαθέτει 5 εφαρμογές.
Η κάθε εφαρμογή αποτελεί ένα αυτοτελές τμήμα της εφαρμογής με τις δικές της λειτουργίες. Συγκεκριμένα έχουμε:

applicant

Πρόκειται για μια εφαρμογή στην οποία έχουμε όλες τις απαραίτητες λειτουργίες για τις ενέργειες των φοιτητών.

internships_app

Στο internships_app έχουμε όλες τις βασικές λειτουργίες που αφορούν τα μοντέλα των χρηστών και τις ενέργειες που σχετίζονται με τις αλλαγές των δεδομένων τους , τις διαγραφές αλλά και τη δημιουργία τους.

secretary

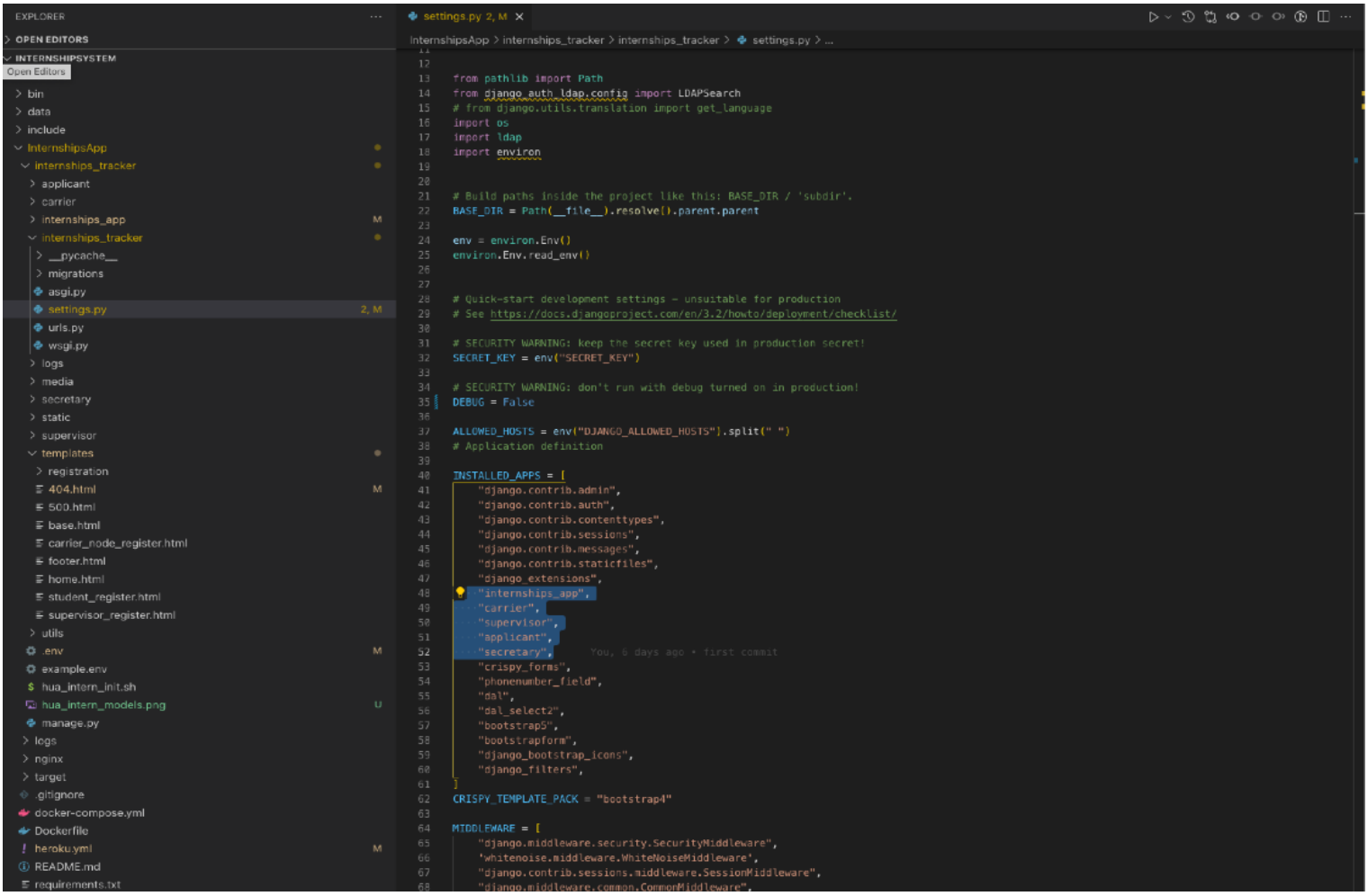
Πρόκειται για την εφαρμογή της γραμματείας στην οποία ορίζονται όλες οι λειτουργίες της.

supervisor

Εφαρμογή για τις λειτουργίες των επιβλέποντων καθηγητών στο σύστημα.

carrier
Εφαρμογή για τις λειτουργίες των Σ.Ε.Φ. και των φορέων.

οι οποίες βρίσκονται στους αντίστοιχους υποφακέλους και έχουν δηλωθεί ως Django apps στο αρχείο settings.py του εμφωλευμένου φακέλου internships_tracker.



settings.py

Ρυθμίσεις στο αρχείο settings.py για τον ορισμό των εφαρμογών της Django Web εφαρμογής.

Η δομή μιας τυπικής Django εφαρμογής αποτελείται από τα εξής βασικά στοιχεία:

views: Τα οποία βρίσκονται στο αρχείο views.py της κάθε εφαρμογής.

models: Βρίσκονται στο αρχείο models.py της κάθε εφαρμογής.

templates: το σύνολο των HTML και JINJA αρχείων που φορτώνει η κάθε εφαρμογή με τη χρήση των views.

migrations: βρίσκεται ο απαραίτητος κώδικας με τον οποίο γίνονται οι αλλαγές στο σχήμα της βάσης σε κάθε deployment(εκκίνηση της εφαρμογής).

urls: Βρίσκονται στο αρχείο urls.py κάθε εφαρμογής, ουτώς ώστε για κάθε

σύνδεσμο(endpoint) να αντιστοιχούμε το κάθε view που εξυπηρετεί τον κάθε χρήστη.

forms: Βρίσκονται στο αρχείο forms.py , στο οποίο περιέχονται διάφορες κλάσεις που αξιοποιούν τα views για τη δημιουργία και την αναπαράσταση μιας φόρμας συμπλήρωσης στοιχείων , αλλά και για την επαλήθευση και τον έλεγχο εγκυρότητας(τύποι δεδομένων , έλεγχοι για μοναδικές εγγραφές , κ.τ.λ.π.).

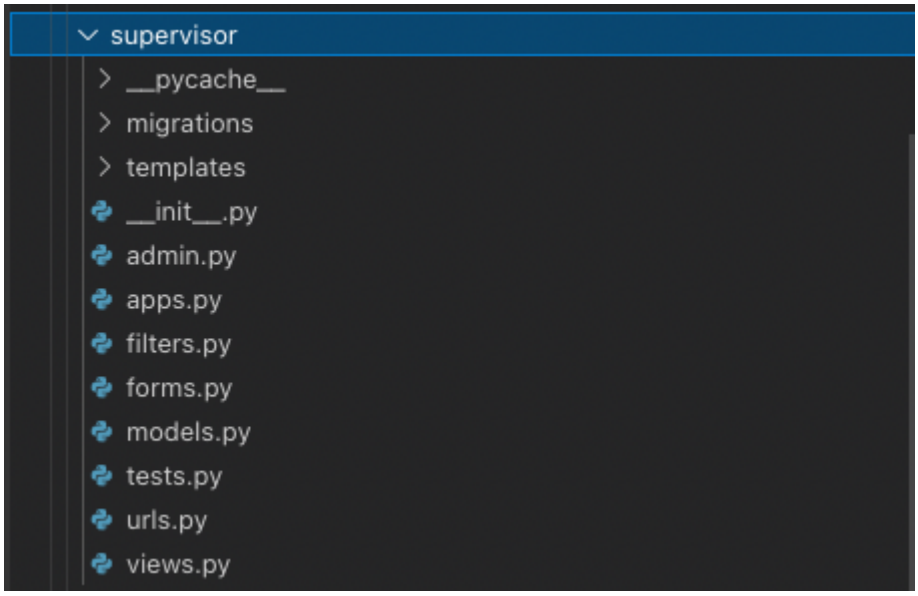
admin: Στο αρχείο admin.py κάθε εφαρμογής βρίσκονται οι ρυθμίσεις για την εισαγωγή μοντέλων της εφαρμογής στο διαχειριστικό περιβάλλον του Django project. Δηλώνουμε ποια αντικείμενα θέλουμε να καταχωρήσουμε στο σύστημα διαχειριστή και με ποιο τρόπο επιθυμούμε να τα προβάλουμε.

mixins: Στο αρχείο mixins.py δηλώνουμε κλάσεις τις οποίες κληρονομούν τα views της εφαρμογής με σκοπό να δώσουμε επιπλέον λειτουργικότητα , όπως με το να κάνουμε υπερφόρτωση στη μέθοδο dispatch() κατά την είσοδο ενός HTTP αιτήματος.

filters: Στο αρχείο filters.py περιλαμβάνονται κλάσεις οι οποίες κληρονομούν το django_filters.FilterSet, με σκοπό τη δημιουργία φίλτρων αναζήτησης σε διάφορα μοντέλα της εφαρμογής.

signals: Στο αρχείο signals.py περιέχονται συναρτήσεις για την δήλωση διάφορων receiver οι οποίοι λαμβάνουν ένα γεγονός όπως η δημιουργία ενός είδους χρήστη και αναλαμβάνουν μια συγκεκριμένη ενέργεια , όπως το να τους τοποθετούν σε ένα συγκεκριμένο γκρουπ χρηστών.

templatetags: Ο φάκελος templatetags περιέχει αρχεία της μορφής (όνομα_εφαρμογής)_tags.py τα οποία έχουν διάφορες συναρτήσεις οι οποίες λαμβάνουν ορίσματα μέσω των templates και εκτελούν κώδικα της python στο backend , όπου μέσω του Jinja template engine μπορούμε να παρουσιάσουμε τις πληροφορίες στο χρήστη με επιθυμητό τρόπο.

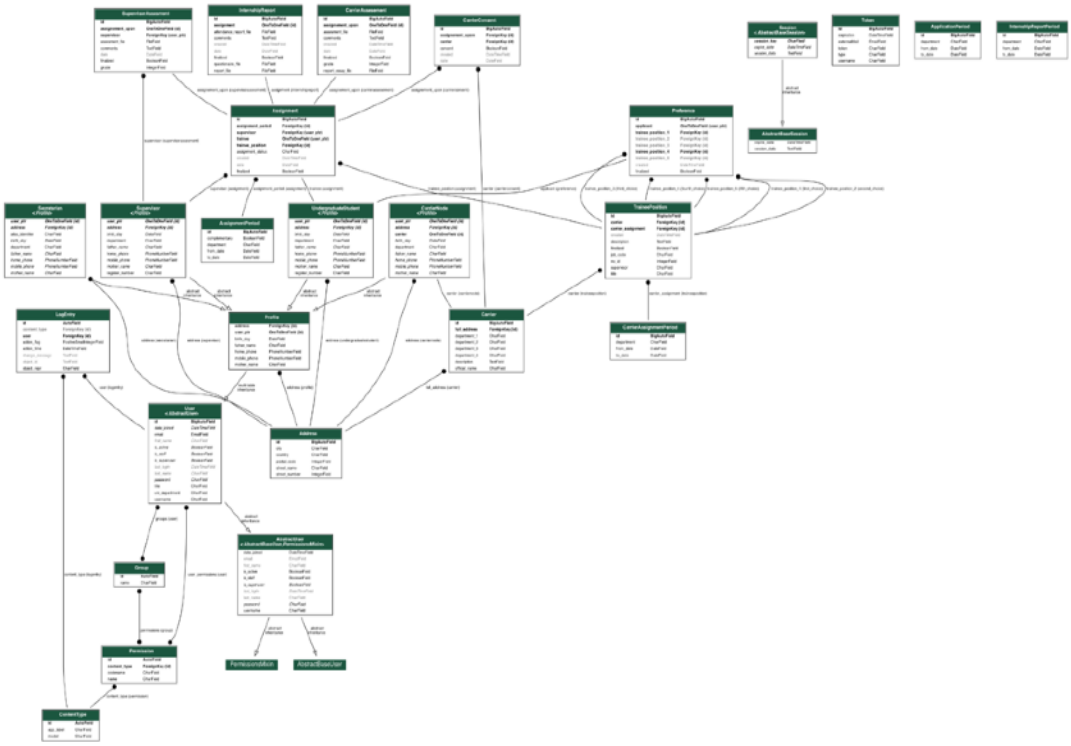


Σύστημα αρχείων Django εφαρμογής

Παράδειγμα δομής αρχείων μιας Django εφαρμογής.

2.5 Μοντέλα της εφαρμογής και σχεδιασμός Βάσης Δεδομένων

2.5.1 Class diagram της εφαρμογής



Class diagram της εφαρμογής

2.5.2 ΧΡΗΣΤΕΣ

Τα μοντέλα των χρηστών του συστήματος κληρονομούν τη κλάση Profile που με τη σειρά του κληρονομεί τη κλάση Abstract User η οποία μας παρέχεται από το Django.

Στο μοντέλο User έχουμε προσθέσει το πεδίο **uni_department** , το οποίο μας χρησιμεύει για τους χρήστες που συνδέονται με ιδρυματικό email του πανεπιστημίου ώστε να προσδιορίσουμε το τμήμα τους. Αυτό το πεδίο αντλείται , κατά τη σύνδεση του χρήστη στον LDAP server μέσω της εφαρμογής.

Το **Profile**(προφίλ) ενός χρήστη περιέχει τα εξής πεδία:

```
   father_name = models.CharField(max_length=255 ,
validators=[alphabetic])
```

```
   mother_name = models.CharField(max_length=255 ,
validators=[alphabetic])
   birth_day = models.DateField()
   address = models.ForeignKey(Address ,
on_delete=models.CASCADE)
   mobile_phone = PhoneNumberField(null=False ,
blank=False , unique=True)
   home_phone = PhoneNumberField(null=False , blank=False
, unique=True)
```

1. **father_name** = Όνομα πατρός τύπου Charfield(συμβολοσειρά από χαρακτήρες , με μέγεθος 255 αλφαβητικούς χαρακτήρες στα λατινικά)

2. **mother_name** = Όνομα μητρός τύπου Charfield(συμβολοσειρά από χαρακτήρες , με μέγεθος 255 αλφαβητικούς χαρακτήρες στα λατινικά)

3. **birth_day** = Ημερομηνία γεννήσεως τύπου ημερομηνίας dd/mm/yyyy(ημέρα/μήνας/χρόνος)

4. **mobile_phone** = Αριθμός κινητής τηλεφωνίας , ο οποίος απαιτεί ως πρόθεμα κωδικό χώρας για την εκχώρηση του π.χ. +30 για ελληνικό αριθμό.

5. **address** = εξωτερικό κλειδί στο Address , το οποίο αναπαριστά τη διεύθυνση
κατοικίας του κάθε χρήστη.

6. **home_phone** = Αριθμός σταθερής τηλεφωνίας , ο οποίος απαιτεί ως πρόθεμα κωδικό χώρας για την εκχώρηση του π.χ. +30 για ελληνικό αριθμό.

Ο **UndergraduateStudent**(προπτυχιακός φοιτητής) κληρονομεί τα πεδία του Profile και έχει τα εξής επιπλέον πεδία:

```
register_number = models.CharField(max_length=10 ,
validators=[alphanumeric])
   department = models.CharField(max_length=3 ,
choices=DEPARTMENT_CHOICES)
```

1. register_number = Αριθμός Μητρώου του κάθε πτυχιακού φοιτητή
2. department = τμήμα φοίτησης του φοιτητή.

Ο **Secretarian** (υπάλληλος γραμματείας) κληρονομεί τα πεδία του Profile.
Ο υπάλληλος της γραμματείας αναλαμβάνει χρέη γραμματείας για 1 ή όλα τα τμήματα του πανεπιστημίου. Μέσω του πεδίου department ορίζουμε το τμήμα ή την επιλογή "ALL" για τη διαχείρισης τους.
Έχει τα εξής επιπλέον πεδία:

```
department =
models.CharField(max_length=3 ,choices=DEPARTMENT_CHOICES
)
alias_identifier = models.CharField(
       max_length=100 , validators=[alphanumeric])
```

1. department = Το τμήμα του Πανεπιστημίου που έχει υπό την επίβλεψη του ο υπάλληλος της γραμματείας. Το τμήμα μπορεί να λάβει τις εξής τιμές:

```
   (IT , "Informatics and Telematics") ,
   (GS , "Geogrpaphy") ,
   (HS , "Economics & Sustainable Development") ,
   (DS , "Nutrition and  Dietetics") ,
   (ALL , "Secreatary") ,
   (NON , "None")
```

2. alias_identifier = απλό αναγνωριστικό για τον υπάλληλο της γραμματείας.

Ο **Supervisor**(επιβλέπων καθηγητής) κληρονομεί τα πεδία του Profile και έχει τα εξής επιπλέον πεδία:

```
register_number =
models.CharField(max_length=10 , validators=[
alphanumeric] , unique=True)
   department = models.CharField(max_length=3 ,
choices=DEPARTMENT_CHOICES)
```

1. department = Τμήμα του επιβλέποντος καθηγητή.
2. register_number = Αριθμός μητρώου καθηγητή.

Ο **CarrierNode** (Σ.Ε.Φ.) , το σημείο επαφής του κάθε φορέα , ο οποίος είναι ο κύριος εκπρόσωπος του φορέα στο σύστημα παρακολούθησης πρακτικών ασκήσεων. Κληρονομεί τα πεδία του Profile και έχει τα εξής επιπλέον πεδία:

```
carrier =
models.OneToOneField(Carrier ,on_delete=models.CASCADE)
department = models.CharField(max_length=150 ,
validators=[alphanumeric])
```

1. carrier = Ξένο κλειδί στην οντότητα Carrier που αναπαριστά το φορέα.
2. department = Τμήμα εργασίας του Σ.Ε.Φ. στο φορέα που εργάζεται.

Carrier

Αναπαράσταση του φορέα, όπου η συμπλήρωση των στοιχείων γίνονται κατά την εγγραφή των Σ.Ε.Φ.

```
 official_name = models.CharField(max_length=120 ,
unique=True)
   full_address = models.ForeignKey(Address ,
on_delete=models.CASCADE)
   description = models.TextField(max_length=1000)
   department_1 = models.CharField(
       max_length=3 , choices=DEPARTMENT_CHOICES_CN
   )  # each carrier is for one department
   department_2 = models.CharField(
       max_length=3 , choices=DEPARTMENT_CHOICES_CN
   )
   department_3 = models.CharField(
       max_length=3 , choices=DEPARTMENT_CHOICES_CN
   )
```

```
    department_4 = models.CharField(
        max_length=3 , choices=DEPARTMENT_CHOICES_CN
    )
```

1. official_name = Επίσημο όνομα φορέα.

2. full_address = Επίσημη διεύθυνση του φορέα , κατά προτίμηση στο κτίριο που διεξάγεται η πρακτική.

3. description = Σύντομη περιγραφή του φορέα.

4. department_1 έως 4 = επιλογές τμημάτων του πανεπιστημίου που ο κάθε φορέας επιθυμεί να διεξάγει πρακτική άσκηση. Κάποια από τις επιλογές μπορεί να λάβει τη τιμή “NON” που υποδηλώνει ότι δεν αντιστοιχεί σε κανένα τμήμα.

2.5.3 ΕΦΑΡΜΟΓΗ ΑΙΤΟΥΝΤΑ (applicant)

Preference:

Προτίμηση είναι ένα μοντέλο που αναπαριστά την αίτηση επιθυμίας για συμμετοχή σε πρακτική άσκηση , όπου ο φοιτητής συμπληρώνει τις θέσεις εργασίας που προτιμά με σειρά προτεραιότητας από τη πρώτη έως τη πέμπτη επιλογή.

```
applicant = models.OneToOneField(
        UndergraduateStudent , on_delete=models.CASCADE)
    trainee_position_1 = models.ForeignKey(
        TraineePosition ,
        related_name="first_choice" ,
        verbose_name="First Choice" ,
        null=False ,
        blank=False ,
        on_delete=models.CASCADE ,
    )
    trainee_position_2 = models.ForeignKey(
        TraineePosition ,
        related_name="second_choice" ,
        verbose_name="Second Choice" ,
        blank=True ,
        null=True ,
        on_delete=models.CASCADE ,
```

```
    )
    trainee_position_3 = models.ForeignKey(
        TraineePosition ,
        related_name="third_choice" ,
        verbose_name="Third Choice" ,
        blank=True ,
        null=True ,
        on_delete=models.CASCADE ,
    )
    trainee_position_4 = models.ForeignKey(
        TraineePosition ,
        related_name="fourth_choice" ,
        verbose_name="Fourth Choice" ,
        null=True ,
        on_delete=models.CASCADE ,
    )
    trainee_position_5 = models.ForeignKey(
        TraineePosition ,
        related_name="fifth_choice" ,
        verbose_name="Fifth Choice" ,
        blank=True ,
        null=True ,
        on_delete=models.CASCADE ,
    )
    finalized = models.BooleanField(default=False)
    created = models.DateTimeField(auto_now_add=True)
```

1. applicant = Προπτυχιακός φοιτητής που συμπληρώνει την αίτηση.

2. trainee_position_1 έως και 5 = Οι θέσεις εργασίας που ενδιαφέρεται για πρακτική άσκηση ο αιτών.

3. finalized = Ένδειξη για το αν η αίτηση είναι οριστικοποιημένη από τη γραμματεία.

4.created = ημερομηνία δημιουργίας της αίτησης.

InternshipReport (Αναφορά ΠΑ)

Η αναφορά πρακτικής άσκησης είναι το συνολικό αρχείο που καταθέτει ο φοιτητής στη γραμματεία μετά το πέρας τη Π.Α.

```
  report_file = models.FileField()
   attendance_report_file = models.FileField()
   questionaire_file = models.FileField()
   assignment = models.OneToOneField(Assignment ,
on_delete=models.CASCADE)
   date = models.DateField(auto_now_add=True)
   finalized = models.BooleanField(default=False)
   comments = models.TextField()
   created = models.DateTimeField(auto_now_add=True)
```

1. **report_file** = Έκθεση πεπραγμένων καθόλη τη περίοδο της Π.Α , αρχείο της μορφής .pdf < 2MB

2. **attendance_report_file** = Παρουσιολόγιο , αρχείο της μορφής .pdf < 2MB.

3. **questionaire_file** = Ερωτηματολόγιο αξιολόγησης φορέα , αρχείο της μορφής .pdf < 2MB.

4. **assignment** = ξένο κλειδί στο μοντέλο Assignment(ανάθεση) , η οποία είναι η τελική ανάθεση που δόθηκε στο φοιτητή.

5. **created** = ημερομηνία δημιουργία της αίτησης ή αλλιώς αποθήκευσης στη Β.Δ.

6. **comments** = Σχόλια σε μορφή κειμένου , που μπορεί να κάνει ο φοιτητής σχετικά με τα υποβληθέντα αρχεία.

2.5.4 ΕΦΑΡΜΟΓΗ ΦΟΡΕΑ(carrier)

Σε αυτή την εφαρμογή έχουν οριστεί οι φάσεις του συστήματος που αφορούν τις ενέργειες των Σ.Ε.Φ. στο σύστημα , αλλά επηρεάζουν και τις λειτουργίες των φοιτητών , καθηγητών και γραμματειών.

Αρχικά έχουμε:

CarrierAssignmentPeriod: Περίοδος δήλωσης ενδιαφέροντος για την Π.Α. από φορείς.

Κατά τη περίοδο αυτή τα Σ.Ε.Φ. εγγράφονται και συμπληρώνουν τα στοιχεία του φορέα τους. Εφόσον εγγραφούν και συνδεθούν επιτυχώς , τότε έχουν τη δυνατότητα να δημιουργήσουν νέες θέσεις εργασίας για το φορέα τους.
Τα πεδία αυτού του μοντέλου είναι τα εξής:

```
   department = models.CharField(unique=True ,
max_length=3 , choices=DEPARTMENT_CHOICES_CN)
   from_date = models.DateField()
   to_date = models.DateField()
```

1. department = ισχύουσα περίοδος για ένα συγκεκριμένο τμήμα του πανεπιστημίου.

2. from_date = ημερομηνία έναρξης της περιόδου.

3. to_date = ημερομηνία λήξης της περιόδου.

ApplicationPeriod: Περίοδος δήλωσης αιτήσεων για φοιτητές.

Στη περίοδο αυτή οι φοιτητές μπορούν να εγγραφούν και να κάνουν τις αιτήσεις τους για τη Π.Α.

Τα πεδία αυτού του μοντέλου είναι τα εξής:

```
department = models.CharField(unique=True ,
max_length=3 , choices=DEPARTMENT_CHOICES_CN)
from_date = models.DateField()
to_date = models.DateField()
```

1.department = ισχύουσα περίοδος για ένα συγκεκριμένο τμήμα του πανεπιστημίου.

2.from_date = ημερομηνία έναρξης της περιόδου.

3.to_date = ημερομηνία λήξης της περιόδου

AssignmentPeriod: Περίοδος αναθέσεων.

Κατά τη περίοδο αναθέσεων οι γραμματείες των τμημάτων λαμβάνουν υπόψη τις προτιμήσεις των φοιτητών και με τα ανάλογα κριτήρια προτείνουν κάποιες αναθέσεις στους φορείς. Τα Σ.Ε.Φ. με τη σειρά τους δίνουν τη συγκατάθεση τους ή δηλώνουν την άρνηση τους για μια υποψήφια ανάθεση.
Σε περίπτωση που υπάρχει συγκατάθεση , η ανάθεση θεωρείται τελική.
Τα πεδία του μοντέλου είναι τα εξής:

```
  department = models.CharField(unique=True ,
max_length=3 , choices=DEPARTMENT_CHOICES_CN)
  from_date = models.DateField()
  to_date = models.DateField()
  complementary = models.BooleanField(default=False)
```

1. department = ισχύουσα περίοδος για ένα συγκεκριμένο τμήμα τουπανεπιστημίου.

2.from_date = ημερομηνία έναρξης της περιόδου.

3. to_date = ημερομηνία λήξης της περιόδου.

4. complementary = ένδειξη για το αν η περίοδος είναι συμπληρωματική.

InternshipReportPeriod: Περίοδος υποβολής αναφορών πρακτικής άσκησης.

Κατά τη διάρκεια της περιόδου αυτής , οι φοιτητές υποβάλλουν τα δικαιολογητικά τους και βαθμολογούνται από το φορέα τους και τον επιβλέπων καθηγητή τους.

```
department = models.CharField(unique=True ,
max_length=3 , choices=DEPARTMENT_CHOICES_CN)
from_date = models.DateField()
to_date = models.DateField()
```

1.department = ισχύουσα περίοδος για ένα συγκεκριμένο τμήμα του πανεπιστημίου.

2. from_date = ημερομηνία έναρξης της περιόδου.

3. to_date = ημερομηνία λήξης της περιόδου

TraineePosition(Θέση πρακτικής άσκησης)

Αναπαράσταση μιας θέσης εργασίας η οποία δημιουργείται από τα Σ.Ε.Φ για τους φορείς Π.Α σε μια συγκεκριμένη περίοδο δήλωσης ενδιαφέροντος για φορείς (Carrier Assignment Period).

```
title = models.CharField(max_length=200)
job_code = models.CharField(max_length=100)
no_id =
models.IntegerField(validators=[MinValueValidator(0) ,
MaxValueValidator(10)])
carrier = models.ForeignKey(Carrier ,
on_delete=models.CASCADE)
carrier_assignment = models.ForeignKey(
       CarrierAssignmentPeriod , on_delete=models.CASCADE
   )
supervisor = models.CharField(max_length=100 ,
validators=[alphabetic])
description = models.TextField(max_length=1500)
created = models.DateTimeField(auto_now_add=True)
finalized = models.BooleanField(default=False)
```

1. title = τίτλος θέσης πρακτικής άσκησης

2. job_code = αναγνωριστικό θέσης εργασίας

3. no_id = σε περίπτωση που έχουμε περισσότερα άτομα για τα οποία προσφέρεται μια θέση εργασίας , τοποθετείται το αναγνωριστικό θέσης εργασίας(job_code) + ένας ακέραιος αριθμός για να δείξουμε πως πρόκειται για μια θέση εργασίας που δέχεται περισσότερα άτομα από ένα , αλλά αφορά τον ίδιο ρόλο.

4.carrier = φορέας εργασίας.

5.carrier_assignment = περίοδος δημιουργίας της θέσης εργασίας.

6. supervisor = συμβολοσειρά για τον υπεύθυνο καθηγητή σε αυτή τη θέση εργασίας

7. description = περιγραφή θέσης εργασίας.

8. finalized = κατάστασης οριστικοποίησης θέσης εργασίας. Εφόσον είναι οριστικοποιημένη από τη γραμματεία , θεωρείται και ανακοινώσιμη για τους φοιτητές του αντίστοιχου τμήματος.

9. created = ημερομηνία δημιουργίας θέσης εργασίας.

Assignment (Ανάθεση)

Το μοντέλο της ανάθεσης είναι η αναπαράσταση μιας ανάθεσης. Πρόκειται για μια ανάθεση ανά φοιτητή για μια συγκεκριμένη περίοδο. Οι αναθέσεις δημιουργούνται από τη γραμματεία με assignment_status (κατάσταση ανάθεσης) = P(Pending) σε αναμονή , μέχρις ότου ληφθεί τελική απόφαση από τους φορείς.
Η γραμματεία για να προωθήσει μια υποψήφια ανάθεση θα πρέπει να έχει οριστικοποιήσει την ανάθεση , εφόσον η τιμή της μεταβλητής finalized γίνει true.

```
class Assignment(models.Model):
    date = models.DateField(auto_now_add=True)
    trainee = models.OneToOneField(
        UndergraduateStudent , unique=True ,
on_delete=models.CASCADE)
    trainee_position = models.ForeignKey(
        TraineePosition , unique=True ,
on_delete=models.CASCADE)
    supervisor = models.ForeignKey(Supervisor ,
on_delete=models.CASCADE)
    assignment_period = models.ForeignKey(
        AssignmentPeriod , on_delete=models.CASCADE)
    finalized = models.BooleanField(default=False)
    assignment_status = models.CharField(
```

```
        max_length=1 , choices=APPLICATION_STATUS ,
default="P")
    created = models.DateTimeField(auto_now_add=True)
```

1.trainee(ασκούμενος) = υποψήφιος ασκούμενος προπτυχιακός φοιτητής

2.created = ημερομηνία δημιουργίας ανάθεσης

3.trainee_position = θέση πρακτικής άσκησης(μοναδική).

4.supervisor = επιβλέπων καθηγητής πρακτικής άσκησης για τη συγκεκριμένη θέση εργασίας.

5.assignment period = περίοδος ανάθεσης που δημιουργήθηκε η συγκεκριμένη ανάθεση.

6.finalized = οριστικοποιημένη ανάθεση , εφόσον η γραμματεία έχει διασφαλίσει πως η πρόταση ανάθεσης είναι η σωστή τότε οριστικοποιεί την ανάθεση και αυτή αποστέλλεται στον αντίστοιχο φορέα ΠΑ.

7.assignment_status = κατάσταση ανάθεσης , οι τιμές που μπορεί να λάβει αυτό το πεδίο είναι P(Pending) σε αναμονή , R(Rejected) απορριμένη , Accepted).Σε περίπτωση που αίτηση έχει κατάσταση R ή A η οποία προήλθε από ενέργεια ενός Σ.Ε.Φ. , τότε θα λάβουμε και ένα αντικείμενο Consent(Συγκατάθεση) που αντιστοιχεί στην ανάθεση(Assignment).

Carrier Consent (Συγκατάθεση φορέα)

Οι φορείς λαμβάνουν τις προτεινόμενες αναθέσεις εργασίας από τις αντίστοιχες γραμματείες και συναινούν ή δηλώνουν την άρνηση τους σε αυτές. Πέρα από το μοντέλο της ανάθεσης , υπάρχει και το αντικείμενο της συγκατάθεσης το οποίο μας δείχνει πότε ένα Σ.Ε.Φ προέβη σε αυτή την ενέργεια.

```
    date = models.DateField(auto_now_add=True)
    carrier = models.ForeignKey(Carrier ,
on_delete=models.CASCADE)
    assignement_upon = models.ForeignKey(Assignment ,
on_delete=models.CASCADE)
    consent = models.BooleanField()
    created = models.DateTimeField(auto_now_add=True)
```

1. created = ημερομηνία δημιουργίας συγκατάθεσης

2.carrier = φορέας

3.assignment_upon = ανάθεση στην οποία έγινε η συγκεκριμένη ενέργεια.

4.consent = απόρριψη/έγκριση ανάθεσης.

CarrierAssesement (Αξιολόγηση του φορέα)

Αξιολόγηση ΠΑ του φοιτητή από το φορέα για μια συγκεκριμένη περίοδο ανάθεσης.

```
assesment_file = models.FileField()
report_essay_file = models.FileField()
date = models.DateField(auto_now_add=True)
assignement_upon = models.OneToOneField(
    Assignment , on_delete=models.CASCADE)
comments = models.TextField(max_length=1000)
finalized = models.BooleanField(default=False)
grade = models.IntegerField(
    validators=[MinValueValidator(0) , MaxValueValidator(10)]
)
created = models.DateTimeField(auto_now_add=True)
```

1.assesement_file = αρχείο αξιολόγησης φοιτητή της μορφής . pdf < 2MB

2.report_essay_file = αρχείο αξιολόγησης έκθεσης πεπραγμένων φοιτητή .pdf < 2MB

3.assignment_upon = ανάθεση στην οποία γίνεται η αξιολόγηση.

4.comments = σχόλια για τη πρακτική άσκηση.

5.finalized = οριστικοποίηση αξιολόγησης από τη γραμματεία.

6.grade = ακέραιος βαθμός από το 0-10

7.created = ημερομηνία δημιουργίας αξιολόγησης.

2.5.5 ΕΦΑΡΜΟΓΗ ΕΠΙΒΛΕΠΟΝΤΟΣ ΚΑΘΗΓΗΤΗ(supervisor)

SupervisorAssesement (Αξιολόγηση επιβλέποντος καθηγητή)

Εφόσον προηγηθεί η αξιολόγηση του φορέα σε μια πρακτική άσκηση , όπου ο συγκεκριμένος καθηγητής είναι επιβλέπων , τότε ο καθηγητής με βάση την αναφορά του φοιτητή και την αξιολόγηση του φορέα δημιουργεί τη δική του αξιολόγηση για το φοιτητή.

```
date = models.DateField(auto_now_add=True)
   assesment_file = models.FileField()
   supervisor = models.ForeignKey(Supervisor ,
on_delete=models.CASCADE)
   assignement_upon = models.OneToOneField(
       Assignment , on_delete=models.CASCADE)
   comments = models.TextField(max_length=1500)
   finalized = models.BooleanField(default=False)
   grade = models.IntegerField(
       validators=[MinValueValidator(0) ,
MaxValueValidator(10)])
```

assesement_file = αρχείο αξιολόγησης φοιτητή της μορφής . pdf < 2MB

assignment_upon = ανάθεση στην οποία γίνεται η αξιολόγηση.

comments = σχόλια για τη πρακτική άσκηση.

finalized = οριστικοποίηση αξιολόγησης από τη γραμματεία.

grade = ακέραιος βαθμός από το 0-10

created = ημερομηνία δημιουργίας αξιολόγησης.

2.6 Διασύνδεση με τον LDAP server του πανεπιστημίου

Η διασύνδεση με τον LDAP του Πανεπιστημίου κρίνεται απαραίτητη για την υλοποίηση αυτού του συστήματος προς τους χρήστες που αποτελούν μέλη της ακαδημαϊκής κοινότητας , και συγκεκριμένα τους καθηγητές και φοιτητές όλων των τμημάτων. Με αυτό τρόπο αυθεντικοποιούνται από τον LDAP server του πανεπιστημίου που διαθέτει όλες τις εγγραφές των μελών του πανεπιστημίου. Αυτό έχει ως κύριο σκοπό , να μη χρειάζεται οι χρήστες να θέτουν νέα credentials για την εισαγωγή τους στο σύστημα αλλά και η αυθεντικοποίηση τους να γίνεται κεντρικά ούτως ώστε να είμαστε σε θέση να γνωρίζουμε τη ταυτότητα του φοιτητή δίχως να υπάρχουν διαδικασίες εγγραφής.

Ωστόσο , για τους σκοπούς αυτής της συνδεσμολογίας ακολουθήθηκε μια σειρά από βήματα για να επιτύχουμε αυτή την επικοινωνία:

1. Στο dockerfile , κατά τη δημιουργία του Ubuntu container στη δεύτερη εντολή , εγκαθιστούμε με τη χρήση του apt τα πακέτα libldap2-dev και ldap-utils.

2.Στο αρχείο requirements.txt κάτω από το root(κύριο) φάκελο χρειαζόμαστε την εξάρτηση **django-auth-ldap==2.2.0** η οποία εγκαθίσταται μέσω του pip σε εντολή που υπάρχει στο dockerfile κατά την εκτέλεση του Ubuntu container.

3.Εν συνεχεία , στο .env αρχείο θέτουμε αυτές τις τιμές όπως στη παρακάτω εικόνα.

```
AUTH_LDAP_SERVER_URI=ldap://ssaml2.hua.gr:389
AUTH_LDAP_BIND_DN=${LDAP_BIND_DN}
AUTH_LDAP_BIND_PASSWORD=${LDAP_BIND_PASS}
AUTH_LDAP_START_TLS=True        You, 5 days ago • first commit
AUTH_LDAP_USER_SEARCH_ATTR=email
AUTH_LDAP_BASE_DN=dc=hua,dc=gr
AUTH_LDAP_GLOBAL_OPTIONS=True
AUTH_LDAP_INTERNAL_DOMAIN=@hua.gr
AUTH_LDAP_GIVEN_NAME=givenName
AUTH_LDAP_SN=sn
AUTH_LDAP_TITLE=title
AUTH_LDAP_DEPARTMENT=schacPersonalPosition
AUTH_LDAP_EMAIL=email
```

Μεταβλητές περιβάλλοντος LDAP

AUTH_LDAP_SERVER_URI απευθύνεται στην web εφαρμογή που ανταποκρίνεται ο ldap server του πανεπιστημίου στη κοινή θύρα 389.

4. Στο αρχείο settings.py θα πρέπει να ορίσουμε τις παρακάτω ρυθμίσεις με σκοπό να επιτεύξουμε ροές αυθεντικοποίησης με τη χρήση του LDAP.

Το AUTH_USER_MODEL αποτελεί το βασικό μοντέλο για το μηχανισμό αυθεντικοποίησης του Django και χρησιμοποιείται για τη δημιουργία χρηστών έπειτα από την επιτυχημένη σύνδεση στον LDAP.

Το AUTHENTICATION_BACKENDS ορίζει δύο flows(ροές) με τα οποία μπορεί να αυθεντικοποιηθεί ένας χρήστης , είτε με email είτε με τη χρήση του LDAP.
Με βάση το σχεδιασμό της εφαρμογής μας , οποιοσδήποτε χρήστης επιχειρεί σύνδεση με email που διαθέτει το επίθεμα “@hua.gr” τότε οδηγείται στον LDAP server και να ολοκληρωθεί η ροή του login(σύνδεσης).

AUTH_ LDAP_USER_ATTR_MAP πρόκειται για ένα χάρτη αντιστοίχισης με τα πεδία που επιστρέφει ο LDAP server και τα πεδία του μοντέλου internships_app_user.

5. Στο αρχείο ldaphandler.py έχουμε όλες τις απαραίτητες λειτουργίες με στη κλάση ldapConnection με σκοπό να:

1. Επικοινωνήσουμε με LDAP server και να πάρουμε απάντηση για το αν υπάρχει χρήστης με συγκεκριμένο email.
2. Να αντλήσουμε στοιχεία μετά την επιτυχή σύνδεση του , όπως ορίσαμε στο AUTH_ LDAP_USER_ATTR_MAP .

Ο κώδικας του αρχείου ldaphandler.py:

```
import ldap
from internships_tracker import settings
from django.contrib.auth import get_user_model

import logging
UserModel = get_user_model()
logger = logging.getLogger('internships_tracker')


def decodeAttribute(entry , attribute):
   if attribute in entry:
       return entry[attribute][0].decode('utf8')
   else:
       return None


def convertToDict(entry):
   return {
       'givenName': decodeAttribute(entry ,
settings.GIVEN_NAME_LDAP_ATTRIBUTE) ,
       'sn': decodeAttribute(entry , settings.SURNAME_LDAP_ATTRIBUTE) ,
       'mail': decodeAttribute(entry ,
settings.EXTERNAL_MAIL_LDAP_ATTRIBUTE) ,
       'title': decodeAttribute(entry , settings.TITLE_LDAP_ATTRIBUTE) ,
       'department': decodeAttribute(entry ,
settings.DEPARTMENT_LDAP_ATTRIBUTE) ,
   }


class ldapConnection:

   handler = None

   def connect(self):
       print("connect")
       """
       Connect to LDAP
       """
       logger.debug('User %s exists' % username)
       print('User %s exists' % username)
       self.handler = ldap.initialize(settings.AUTH_LDAP_SERVER_URI)
       if settings.AUTH_LDAP_START_TLS:
           self.handler.start_tls_s()
```

```
        some = self.handler.simple_bind(
            settings.AUTH_LDAP_BIND_DN , settings.AUTH_LDAP_BIND_PASSWORD)
        print("details about the connection with ssaml2" , some)

    def verify(self , uid , externalMail):
        print("verify")
        """
        Verify that user externalMail is correct
        """
        query = '(&(' + settings.EXTERNAL_MAIL_LDAP_ATTRIBUTE + \
            '=' + externalMail + ')(uid=' + uid + '))'
        print("query is here" , query)
        result = self.handler.search_s(
            settings.AUTH_LDAP_BASE_DN , ldap.SCOPE_SUBTREE , query)
        print("here is the result" , result)
        if len(result) == 1:
            return True
        else:
            return False

    def getUserMail(self , uid):
        print("getUserMail")
        """
        get user externalMail
        """
        print("get external mail" , self.request)
        query = '(uid=%s)' % uid
        result = self.handler.search_s(
            settings.AUTH_LDAP_BASE_DN , ldap.SCOPE_SUBTREE , query)
        if len(result) == 1:
            ldapEntry = result[0][1]
            if settings.EXTERNAL_MAIL_LDAP_ATTRIBUTE in ldapEntry:
                return ldapEntry[settings.EXTERNAL_MAIL_LDAP_ATTRIBUTE]
[0].decode('utf8')
            else:
                return None
        else:
            return None

    def getUserInfo(self , uid):
        print("getUserInfo")
        """
        get user information: username , givenName , sn and externalMail
        """
        query = '(uid=%s)' % uid
        print("the user is " , uid)
```

```
        result = self.handler.search_s(
            settings.AUTH_LDAP_BASE_DN , ldap.SCOPE_SUBTREE , query)
        print("result of ldap search" , result)
        if len(result) == 1:
            some = convertToDict
            print("here are some details about the user" , some.__dict__)
            return convertToDict(result[0][1])
        else:
            return None

    def close(self):
        self.handler.unbind_s()
```

Σε γενικές γραμμές αυτά ήταν τα βήματα που ακολουθήθηκαν για τη διασύνδεση με τον LDAP server του πανεπιστημίου.

2.7 Views

Τα Views θα λέγαμε πως αποτελούν το μέρος στο οποίο τοποθετούμε τη "λογική" των εφαρμογών μας. Στηρίζεται στην ύπαρξη ενός ή πολλών μοντέλων(models) καθώς και επεξεργάζεται τα δεδομένα με συγκεκριμένο τρόπο , με τελικό σκοπό να περάσουν στα templates και ο χρήστης να λάβει την ανάδραση την οποία περιμένει. Στο σύστημα αυτό , χρησιμοποιήσαμε αρκετά τα Class-based views κληρονομώντας τα generic views του Django.Σε γενικές γραμμές , τα generic views και κατ επέκταση τα CBVs μας βοηθούν να παράγουμε views χωρίς επαναλαμβανόμενο(boilerplate) κώδικα.

Generic views που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής:

CreateView : Πρόκειται για ένα CBV του οποίου ο στόχος είναι η αποθήκευση αντικειμένων στη Β.Δ. .Η κλάση του αντικειμένου ορίζεται στο model , τα πεδία της φόρμας που αποτελούν κομμάτι εισόδου για τη συμπλήρωση των στοιχείων από το χρήστη ορίζονται στα forms του Django και μέσω των αντίστοιχων templates που ορίζουμε προς φόρτωση σε αυτό το view αυτό παραδίδουμε την αντίστοιχη φόρμα στο τελικό χρήστη.Εφόσον ο χρήστης συμπληρώσει επιτυχώς τα δεδομένα μπορούμε να αποθηκεύσουμε τα κατάλληλα δεδομένα που αφορούν ένα συγκεκριμένο μοντέλο.

UpdateView: Στόχος του view αυτού , είναι η επεξεργασία ενός υπάρχοντος μοντέλου που αντιστοιχεί σε κάποια εγγραφή στη βάση δεδομένων. Ορίζουμε τη

κλάση του μοντέλου στο view , και τη φόρμα στην οποία πρέπει να βασιστούμε για τη συμπλήρωση των στοιχείων του.
Η ανάκτηση γίνεται με βάση το id του αντικειμένου που βρίσκεται στο url της φόρμας , που φορτώνεται το τελικό template για το αντικείμενο προς επεξεργασία.
Εφόσον γίνει η ανάκτηση των δεδομένων μέσω της φόρμας μπορούμε να φορτώσουμε το κατάλληλο template με προσυμπληρωμένα τα στοιχεία του μοντέλου για τυχόν επεξεργασία.Εν τέλει , ο χρήστης κάνει τις επιθυμητές τροποποιήσεις και το μοντέλο αποθηκεύεται εκ νέου στη Β.Δ. .

DetailView: Αποσκοπεί στην απεικόνιση των αντικειμένων που είναι αποθηκευμένα στη Β.Δ. .Ορίζουμε στο view αυτό τη κλάση του αντικειμένου το οποίο θέλουμε να προβάλουμε. Η ανάκτηση του αντικειμένου γίνεται με βάση το όρισμα id , που βρίσκεται στο url στο οποίο αιτούμαστε την προβολή αυτού του αντικειμένου.

ListView: Εξυπηρετεί στην αναπαράσταση πολλαπλών αντικειμένων/μοντέλων ίδιου τύπου.Ορίζουμε στο view το μοντέλο των αντικειμένων που θέλουμε να προβάλουμε.

2.7.1 ΕΦΑΡΜΟΓΗ ΑΙΤΟΥΝΤΑ(applicant)

PrerenceUpdateView(UpdateView):Μέσω αυτού του view για όσο το **ApplicationPeriod** είναι ενεργό ο φοιτητής μπορεί να ενημερώνει την αίτηση δήλωσης ενδιαφέροντος για Π.Α.

PrerenceDetailView(DetailView):Μέσω αυτού του view για όσο το **ApplicationPeriod** είναι ενεργό ο φοιτητής μπορεί να δει τα στοιχεία της αίτησης δήλωσης ενδιαφέροντος για την Π.Α. με τις επιλογές των θέσεων εργασίας του.

CreatePreferenceView(CreateView):Μέσω αυτού του view για όσο το Application Period είναι ενεργό ο φοιτητής μπορεί να δημιουργήσει μια αίτηση δήλωσης ενδιαφέροντος για την Π.Α. συμπληρώνοντας τις θέσεις που τον ενδιαφέρουν από τη πρώτη έως και τη πέμπτη επιλογή για τις οριστικοποιημένες θέσεις Π.Α. στο τμήμα του.

TraineePositionStudentListView(ListView): Μέσω αυτού του view για όσο το **ApplicationPeriod** είναι ενεργό ο φοιτητής μπορεί να δει το σύνολο των οριστικοποιημένων θέσεων εργασίας σε μια σελίδα.

MyCarrierConsentDetailView(DetailView): Ο φοιτητής μπορεί να δει την ανάθεση του , με τα στοιχεία της θέσης εργασίας του εφόσον ληφθεί η συγκατάθεση του φορέα και η ανάθεση θεωρηθεί τελική.

InternshipReportView(CreateView): Ο φοιτητής για όσο είναι ενεργό το **InternshipReportPeriod** μπορεί να ανεβάσει τα αρχεία που είναι απαραίτητα για την αξιολόγηση του.

InternshipUpdateView(UpdateView): Ο φοιτητής για όσο είναι ενεργό το **InternshipReportPeriod** μπορεί να αλλάξει τα αρχεία που έχει προηγουμένως ανεβάσει στη πλατφόρμα έως ότου η γραμματεία οριστικοποιήσει το αρχείο του.Σε περίπτωση οριστικοποιήσης τότε έχει δικαίωμα προβολής της αίτησης και όχι επεξεργασίας.

2.7.2 ΕΦΑΡΜΟΓΗ ΓΡΑΜΜΑΤΕΙΑΣ(secretary)

ApprovalRejectionUndergratuateStudentListView(ListView): Σε αυτό το view ο υπάλληλος της γραμματείας βλέπει τις εγγραφές των φοιτητών του τμήματος που έχουν εγγραφεί ή αναμένουν την εγγραφή τους. Μπορεί να δει τα στοιχεία τους όπως όνοματεπώνυμο, αριθμό μητρώου, ημερομηνία εγγραφής , αν εκκρεμεί η εγγραφή του κ.τ.λ.π.

student_approval_rejection: Ο υπάλληλος της γραμματείας μπορεί να δει την εγγραφή ενός φοιτητή και σε περίπτωση που εκκρεμεί η εγγραφή του έχει τη δυνατότητα να πατήσει το κουμπί της ενεργοποίησης του χρήστη.

student_approve: Εφόσον ο υπάλληλος της γραμματείας ελέγξει πως ένα φοιτητής πληροί τα κριτήρια για να συμμετάσχει στο πρόγραμμα Π.Α. ενεργοποιεί τον φοιτητή πατώντας το κουμπί της αποδοχής στο template που έχει φορτωθεί από το view:**student_approval_rejection.**

student_reject: Εφόσον στο template που φορτώθηκε από το **student_approval_rejection** έχει δοθεί εντολή για απόρριψη του φοιτητή τότε οδηγούμαστε στο view: **student_reject** όπου η εγγραφή του φοιτητή αφαιρείται από τη ΒΔ.

ApprovalRejectionCarrierNodeListView(ListView): Σε αυτό το view ο υπάλληλος της γραμματείας βλέπει τα Σ.Ε.Φ. με τους φορείς τους που έχουν κάνει εγγραφή στο σύστημα. Σε μορφή λίστας βλέπουν όλους τα Σ.Ε.Φ. , με τα ονοματεπώνυμα των εκπροσώπων , το όνομα του φορέα , στοιχεία επικοινωνίας κ.τ.λ.π.

carrier_node_approval_rejection: Ο υπάλληλος της γραμματείας μπορεί να δει την εγγραφή ενός Σ.Ε.Φ. αλλά και του φορέα του. Σε περίπτωση που εκκρεμεί η εγγραφή του έχει τη δυνατότητα να πατήσει το κουμπί της ενεργοποίησης του χρήστη.

carrier_node_approve: Εφόσον ο υπάλληλος της γραμματείας ελέγξει πως ένα Σ.Ε.Φ. πληροί τα κριτήρια για να συμμετάσχει στο πρόγραμμα Π.Α. ενεργοποιεί το Σ.Ε.Φ πατώντας το κουμπί της αποδοχής στο template που έχει φορτωθεί από το **carrier_node_approval_rejection.**

carrier_node_reject: Εφόσον στο template που φορτώθηκε από το **carrier_node_approval_rejection** έχει δοθεί εντολή για απόρριψη του φοιτητή τότε οδηγούμαστε στο view: **carrier_node_reject** όπου η εγγραφή του Σ.Ε.Φ αφαιρείται από τη Β.Δ. μαζί με το φορέα.

ApprovalTraineePositionsListView(ListView): Ο υπάλληλος της γραμματείας βλέπει το σύνολο των θέσεων που δημιουργούνται από τους φορείς για το τμήμα τους. Σε κάθε θέση εργασίας βλέπουν τα στοιχεία του φορέα , τη περιγραφή της θέσης εργασίας , τη ημερομηνία δημιουργίας , το αναγνωριστικό(κωδικό) εργασίας κ.τ.λ.π.

trainee_position_approval_rejection: Ο υπάλληλος της γραμματείας μπορεί να δει τα στοιχεία μιας θέσης εργασίας που αιτείται ένας φορέας κατά το CarrierAssignmentPeriod και να την απορρίψει ή να την εγκρίνει.

SecretaryTraineePositionUpdateView(UpdateView): Η γραμματεία μπορεί να επεξεργαστεί τα στοιχεία της αίτησης , όπως τον τίτλο και το κωδικό εργασίας όταν αυτό κριθεί απαραίτητο(σε περίπτωση κωλύματος ή τυπογραφικού λάθους).

trainee_position_approve: Έγκριση θέσης στο πρόγραμμα Π.Α. για το τρέχον χρονικό διάστημα.

SecretaryTraineePositionDeleteView(DeleteView): Διαγραφή θέσης πρακτικής άσκησης. Κατά την επισκόπηση των στοιχείων μιας υποψήφιας θέσης Π.Α. ο φορέας απορρίπτει το αίτημα του φορέα με αποτέλεσμα να διαγράφεται από το σύστημα η θέση εργασίας.

ApprovalPreferencesListView(ListView): Προβολή όλων των αιτήσεων ενδιαφέροντος για Π.Α. από τους φοιτητές του τμήματος στο οποίο ανήκει η γραμματεία.

SecretaryPreferenceUpdateView(UpdateView): Ενημέρωση αιτήσεων ενδιαφέροντος για Π.Α από φοιτητές σε περίπτωση λάθος συμπλήρωσης κατόπιν επικοινωνίας με το φοιτητή.

preference_approval_rejection: Προβολή αίτησης ενδιαφέροντος για Π.Α. από ένα συγκεκριμένο φοιτητή.

AssignmentListView(ListView): Προβολή οριστικοποιημένων και υποψήφιων αναθέσεων.

AssignmentCreateView: Δημιουργία υποψήφιας ανάθεσης. Πρόκειται για draft(πρόχειρη) εγγραφή , η οποία θα πρέπει να οριστικοποιηθεί από τη γραμματεία για να διαδοθεί στο φορέα Π.Α. .

AsssignmentDetailView: Προβολή μεμονωμένης υποψήφιας/οριστικοποιημένης/ πρόχειρης ανάθεσης.

AsssignmentDetailView: Ενημέρωση στοιχείων υποψήφιας/οριστικοποιημένης/ πρόχειρης ανάθεσης.

AssignmentDeleteView: Διαγραφή υποψήφιων/οριστικοποιημένων/πρόχειρων αναθέσεων. Τα consents στη περίπτωση οριστικοποιημένων αναθέσεων παραμένουν στη Β.Δ. με σκοπό να υπάρχει ιστορικό απόδοσης αναθέσεων. Στη περίπτωση που πρέπει να δημιουργεί νέα ανάθεση για την ίδιο φοιτητή , προτείνεται να διαγράφεται το αντικείμενο **Consent** από τον **Admin** του συστήματος.

assignment_approve: Οριστικοποίηση της ανάθεσης από τη γραμματεία εφόσον ληφθεί η συγκατάθεση από το Σ.Ε.Φ. για μια ανάθεση.

assignment_reject: Ορισμός μιας ανάθεσης ως απορημένης από τη γραμματεία.

InternshipsReportListView(ListView): Προβολή αρχείων αξιολόγησης φοιτητών για το τμήμα του υπαλλήλου της γραμματείας.

internship_report_finalize: Οριστικοποίηση αρχείου αξιολόγησης φοιτητή. Το αρχείο είναι έτοιμο προς προβολή από τους φορείς.

internship_report_discard: Απόρριψη αρχείου αξιολόγησης που έχει υποβληθεί από το φοιτητή.

CarrierAssesementListView(ListView): Προβολή όλων των αρχείων αξιολόγησης των φορέων , οριστικοποιημένων και μη.

CarrierAssesementDetailView(DetailView): Προβολή αξιολόγησης Π.Α. ενός φοιτητή για μια θέση εργασίας από τον φορέα.

carrier_assesement_finalize: Οριστικοποίηση αξιολόγησης φορέα από τη γραμματεία για ένα φοιτητή σε μια θέση Π.Α. .Έπειτα από αυτή την ενέργεια το συνολικό αρχείο αξιολόγησης από το φοιτητή και τον φορέα κοινοποιείται στον επιβλέποντα καθηγητή.

carrier_assesement_discard: Ορισμός αξιολόγησης φορέα για Π.Α. ενός φοιτητή ως ανεπαρκής ή λανθασμένη.

SupervisorAssesementListView(ListView): Προβολή όλων των αρχείων αξιολόγησης των επιβλέποντων καθηγητών , οριστικοποιημένων και μη.

SupervisorAssesementDetailView(DetailView): Προβολή αξιολόγησης Π.Α. ενός φοιτητή για μια θέση εργασίας από τον καθηγητή.

supervisor_assesement_finalize: Οριστικοποίηση αξιολόγησης επιβλέποντα καθηγητή από τη γραμματεία για ένα φοιτητή σε μια θέση Π.Α. .Έπειτα από αυτή την ενέργεια , ο τελικός βαθμός του φοιτητή μπορεί να μεταφερθεί στο φοιτητολόγιο.

supervisor_assesement_discard: Ορισμός αξιολόγησης επιβλέποντα καθηγητή για Π.Α. ενός φοιτητή ως ανεπαρκής ή λανθασμένη

2.7.3 ΕΦΑΡΜΟΓΗ ΕΠΙΒΛΕΠΟΝΤΩΝ ΚΑΘΗΓΗΤΩΝ(secretary)

AssignmentListView(ListView):Μέσω αυτού του view ο καθηγητής βλέπει τις αναθέσεις φοιτητών που έχουν αναλάβει θέση εργασίας στην οποία είναι επιβλέπων καθηγητής , οι οποίες είναι τελικές. Τελική ανάθεση ορίζεται οποιαδήποτε ανάθεση έχει λάβει συγκατάθεση από το φορέα στον οποίο βρίσκεται η εργασία.

AsssignmentDetailView(DetailView): Ο καθηγητής επιλέγει μια ανάθεση και βλέπει τα στοιχεία του ασκούμενου φοιτητή(πληροφορίες για το φοιτητή και στοιχεία επικοινωνίας) , όπως και τη θέση της Π.Α. του και τον φορέα της Π.Α.

SupervisorAssesementCreateView(CreateView):Ο καθηγητής μπορεί να δημιουργήσει μια αξιολόγηση για φοιτητές που έχουν αναλάβει θέση εργασίας στην οποία είναι επιβλέπων καθηγητής για τη συγκεκριμένη περίοδο αναφοράς πρακτικής άσκησης(InternshipReportPeriod) και συμπληρώνει όλα τα πεδία που αφορούν την αξιολόγηση του φοιτητή πάνω σε μια ανάθεση.

SupervisorAssesementListView(ListView): Ο καθηγητής μπορεί να δει το σύνολο των αξιολογήσεων αναθέσεων Π.Α. που έχει κάνει στο παρελθόν.

SupervisorAssesementDetailView(DetailView): Ο καθηγητής μπορεί να επιλέξει μια αξιολόγηση Π.Α. και να δει τα στοιχεία του ασκούμενου , της εργασίας καθώς και τα στοιχεία που αφορούν την αξιολόγησή του , όπως η αναφορά πρακτικής άσκησης και την αξιολόγηση του φορέα αλλά και τη δική του αξιολόγηση.

SupervisorAssesementUpdateView(UpdateView): Ο καθηγητής μπορεί να ενημερώσει τα στοιχεία της αξιολόγησης του για μια Π.Α., όπως να ανεβάσει ένα διορθωμένο αρχείο ή να αλλάξει τη βαθμολογία.

2.7.4 ΕΦΑΡΜΟΓΗ ΦΟΡΕΑ(carrier)

TraineePositionListView(ListView): Το Σ.Ε.Φ. μπορεί να δει το σύνολο των θέσεων που έχει υποβάλλει στο σύστημα καθώς και διάφορα στοιχεία που έχει συμπληρώσει όπως ο τίτλος , η περιγραφή , αναγνωριστικό θέσης εργασίας αλλά και την εξέλιξη του αιτήματος του.

TraineePositionCreateView(CreateView): Το Σ.Ε.Φ. μέσω αυτού view , μπορεί να δημιουργήσει μια νέα θέση εργασίας για ένα από τα τμήματα του πανεπιστημίου που απευθύνεται ο φορέας του όταν το CarrierAssignmentPeriod είναι ενεργό.

TraineePositionDetailView(DetailView): Το Σ.Ε.Φ μπορεί να δει τα στοιχεία μιας υποβληθείσας θέσης εργασίας μεμονωμένα.

TraineePositionUpdateView(UpdateView): Το Σ.Ε.Φ. μπορεί να τροποποιήσει τα στοιχεία της αίτησης του , όσο το CarrierAssignmentPeriod είναι ενεργό για το τμήμα του πανεπιστημίου που αναρτά τη θέση αυτή και υπό τη προϋπόθεση πως δεν έχει παρθεί τελική απόφαση από τη γραμματεία.

TraineePositionDeleteView(DeleteView): Το Σ.Ε.Φ κατά το CarrierAssignmentPeriod μπορεί να διαγράψει μια υποβεβλημένη θέση εργασίας , υπό τη προϋπόθεση πως δεν έχει παρθεί τελική απόφαση από τη γραμματεία.

carrier_assignment_not_found: Σε περίπτωση που το Σ.Ε.Φ προσπαθήσει να ανακατευθυνθεί σε μια σελίδα όπου υπάρχουν περιορισμοί με βάση το CarrierAssignmentPeriod και βρίσκεται εκτός του διαθέσιμου χρονικού διαστήματος, το view αυτό αναλαμβάνει να φορτώσει μια σελίδα με ένα μήνυμα λάθους

AssignmentListView(ListView): Το Σ.Ε.Φ βλέπει τη λίστα με όλες τις υποψήφιες αναθέσεις των γραμματειών σχετικά με τις θέσεις πρακτικής άσκησης που έχουν υποβάλλει στο σύστημα. Προϋπόθεση είναι να ο φοιτητής να μην έχει άλλη ανάθεση και το Assignment period να είναι ενεργό για το τμήμα του ασκούμενου φοιτητή.

assignment_accept: Αποδοχή ανάθεσης για έναν υποψήφιο ασκούμενο φοιτητή για μια θέση Π.Α. . Προϋπόθεση είναι να ο φοιτητής να μην έχει άλλη ανάθεση και το Assignment period να είναι ενεργό για το τμήμα του ασκούμενου φοιτητή.

assignment_reject: Άρνηση Σ.Ε.Φ. σε πρόταση ανάθεσης για Π.Α. για ένα συγκεκριμένο φοιτητή για μια συγκεκριμένη θέση.

AcceptedAssignmentsListView(ListView): Λίστα με τους ασκούμενους φοιτητές του φορέα , ή αποδεκτές προτάσεις ανάθεσης Π.Α.

AcceptedAssignmentDetailView(DetailView): Προεπισκόπηση ενός ασκούμενου φοιτητή.

CarrierAssesementCreateView(CreateView): Εφόσον το InternshipsReportPeriod είναι ενεργό και εφόσον έχει γίνει αποδεκτό το αρχείο της αξιολόγησης του φοιτητή από την αρμόδια γραμματεία , το Σ.Ε.Φ μπορεί να δημιουργήσει μια αξιολόγηση για τον ασκούμενο φοιτητή.

CarrierDetailView(DetailView): Σε αυτό το view το Σ.Ε.Φ. μπορεί να δει τα στοιχεία του φορέα του.

CarrierUpdateView(UpdateView): Σε αυτό το view το Σ.Ε.Φ μπορεί να αλλάξει τα στοιχεία του φορέα , όπως το όνομα τη διεύθυνση κ.τ.λ.π.. Σε περίπτωση που θέλει να προσθέσει επιπλέον τμήματα του πανεπιστημίου πρέπει να απευθυνθεί στη γραμματεία του πανεπιστημίου για αυτή την ενέργεια.

ΚΕΦ.3 ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

3.1 ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ

Σε αυτό το κεφάλαιο θα αναλύσουμε το τρόπο με τον οποίο εγκαθιστούμε και θέτουμε σε εφαρμογή το Σύστημα παρακολούθησης Π.Α.

Προαπαιτούμενα για την επιτυχή εγκατάσταση της εφαρμογής είναι τα εργαλεία:

1.Git

2.Docker
3.docker-compose

Αν πατήσετε στα παραπάνω εργαλεία θα ανακατευθυνθείτε στους κατάλληλους συνδέσμους που υπάρχουν οι οδηγίες για την εγκατάσταση τους στις μεγαλύτερες πλατφόρμες που υποστηρίζονται.

Το αποθετήριο κώδικα για την εφαρμογή βρίσκεται στο σύνδεσμο Internships Tracker. Για να εγκαταστήσετε τοπικά την εφαρμογή , πρέπει να τρέξετε την εντολή:

git clone https://github.com/ipanagiotopoulos/InternshipsTracker

Εφόσον αποκτήσετε το πηγαίο κώδικα της εφαρμογής , μπορείτε πλέον να πλοηγηθείτε στο σύστημα αρχείων της εφαρμογή κάτω από το φάκελο που έχετε τον ".git" φάκελο.

για να δείτε αν υπάρχει ο φάκελος αυτός , μπορείτε να εκτελέσετε την εντολή:

ls -a

Στον φάκελο internships_tracker θα εντοπίσετε ένα αρχείο example.env , σε αυτό έχουν οριστεί κάποιες μεταβλητές περιβάλλοντος για το deployment της εφαρμογής μας. Ωστόσο αυτές οι εντολές θα πρέπει να μεταφερθούν στο αρχείο .env , οπότε θα μεταφέρουμε τα περιεχόμενα του αρχείου "example.env" στο ".env" με τη χρήστη της εντολής:

cp example.env .env

Εφόσον έχει ολοκληρωθεί η αντιγραφή , θα πρέπει να συμπληρωθούν κάποιες μεταβλητές περιβάλλοντος που αφορούν το περιβάλλον διαχειριστή και τη Β.Δ.

DB_DATABASE_NAME
DB_USERNAME
DB_PASSWORD

Στο **DB_DATABASE_NAME** ορίζουμε το όνομα της βάσης

Στο **DB_USERNAME** ορίζουμε όνομα χρήστη για τον διαχειριστή της βάσης

Στο **DB_PASSWORD** το συνθηματικό του διαχειριστή της βάσης

DB_HOST=db
DB_PORT=5432

Στα παραπάνω δύο πεδία δε προτείνουμε στον χρήστη να μη προβεί σε αλλαγές, ειδάλλως θα χρειαστούν αλλαγές στα αρχεία docker-compose.yml , Dockerfile και hua_intern_init.sh.

Για το περιβάλλον διαχειριστή του Django (Django admin) , πρέπει να συμπληρωθούν οι εξής μεταβλητές περιβάλλοντος:

DJANGO_SUPERUSER_PASSWORD=

DJANGO_SUPERUSER_USERNAME=

DJANGO_SUPERUSER_EMAIL=

Στο **DJANGO_SUPERUSER_PASSWORD** ορίζουμε το κωδικό για τον υπερχρήστη του Django Admin.

Στο **DJANGO_SUPERUSER_USERNAME** ορίζουμε όνομα χρήστη για τον υπερχρήστη στο Django Admin.

Στο **DJANGO_SUPERUSER_EMAIL** ορίζουμε την ηλεκτρονική ταχυδρομική διεύθυνση του υπερχρήστη στο Django Admin.

Εν συνεχεία , ορίστε τη μεταβλητή **DJANGO_ALLOWED_HOSTS=localhost** ώστε να τρέξετε την εφαρμογή σας τοπικά.

Σε περίπτωση που έχετε κάποιον reverse-proxy ή διακομιστή με ssl πιστοποιητικό, ορίστε το FQDN της εφαρμογής σας.

Στο αρχείο settings.py μπορείτε να αλλάξετε τη μεταβλητή DEBUG από True σε False αν θέλετε να τρέξετε την εφαρμογή σας παραγωγικό περιβάλλον. Σε δοκιμαστικό ή development περιβάλλον αφήνετε την τιμή True.

Στη συνέχεια για να εκτελέσετε την εφαρμογή , τρέξτε την εντολή:

docker-compose up –build -d

Η παράμετρος/flag -d, σημαίνει πως η εφαρμογή τρέχει στο παρασκήνιο δίχως να έχουμε στη κονσόλα μας ζωντανά τα logs της εφαρμογής.

Για να βεβαιωθούμε πως η εφαρμογή τρέχει σωστά , τα εξής 3(τρία) containers πρέπει να τρέχουν:

internships_web: Ubuntu container με την Django εφαρμογή.

nginx: διακομιστής nginx.

internships_db: βάση δεδομένων της PostgreSQL.

Για να βεβαιωθούμε ότι τα containers είναι ενεργά μπορούμε να τρέξουμε την εντολή:
docker ps

Εφόσον τα παραπάνω ισχύουν , μπορούμε να ξεκινήσουμε να χρησιμοποιούμε την εφαρμογή , πηγαίνοντας στο localhost στο πρόγραμμα περιήγησης που διαθέτουμε.

Στο localhost/admin, έχουμε πρόσβαση στο περιβάλλον διαχειριστή.

Για περισσότερες οδηγίες, μπορείτε να επισκεφθείτε τον σύνδεσμο:
https://github.com/ipanagiotopoulos/InternshipsTracker#readme

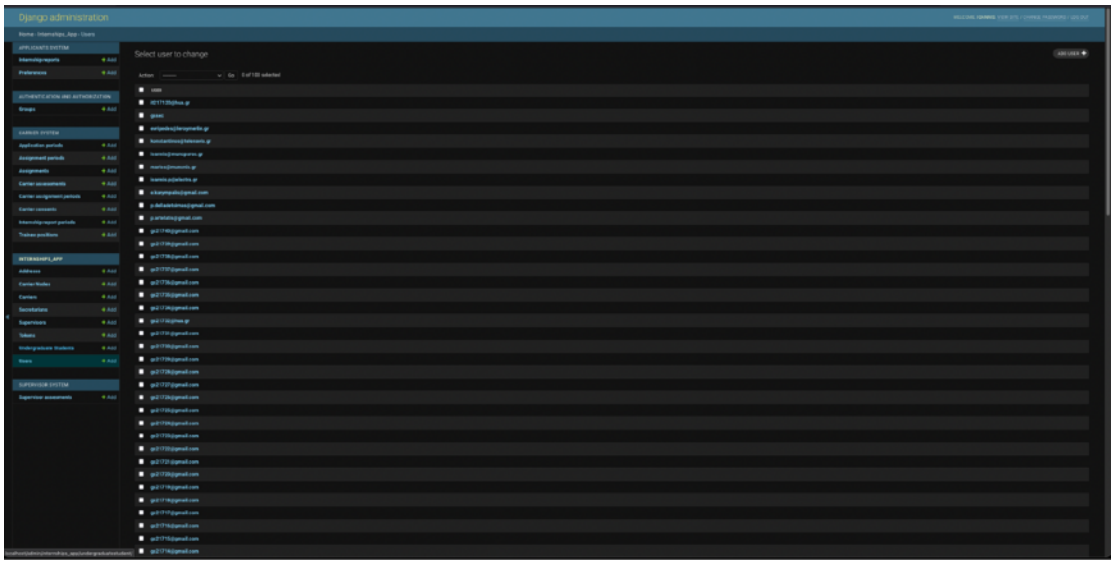
3.2 Περιβάλλον Admin.

Το περιβάλλον Admin(διαχειριστή) μας δίνει πρόσβαση σε όλα τα κρίσιμα δεδομένα της εφαρμογής, που αποτελούν μέρος ή ακόμα και ολόκληρο το σχήμα της Β.Δ. που αξιοποιεί το σύστημα.

Ο Admin είναι υπεύθυνος για τη διατήρηση των δεδομένων και τη διασφάλιση της ομαλής λειτουργίας του συστήματος. Πρόκειται για ένα τεχνικό χρήστη ο οποίος καταλαβαίνει σε ικανοποιητικό βαθμό πως τα δεδομένα και οι σχέσεις τους με τις

λειτουργικότητες της εφαρμογής μπορούν να επηρεάσουν τη διεξαγωγή των φάσεων του συστήματος.

Κατά κύριο λόγο , στο admin περιβάλλον έχουμε πρόσβαση στο σύνολο χρηστών του συστήματος και πιο συγκεκριμένα σε αυτούς με το ρόλο του φοιτητή , του υπαλλήλου της γραμματείας , του Σ.Ε.Φ και του επιβλέποντα καθηγητή. Για κάθε μια κατηγορία διαθέτουμε και ένα ListView κάτω από την εφαρμογή "IntenrnshipsApp" όπου μπορούμε να προβούμε σε ενέργειες όπως η διαγραφή και η επεξεργασία των στοιχείων των χρηστών.

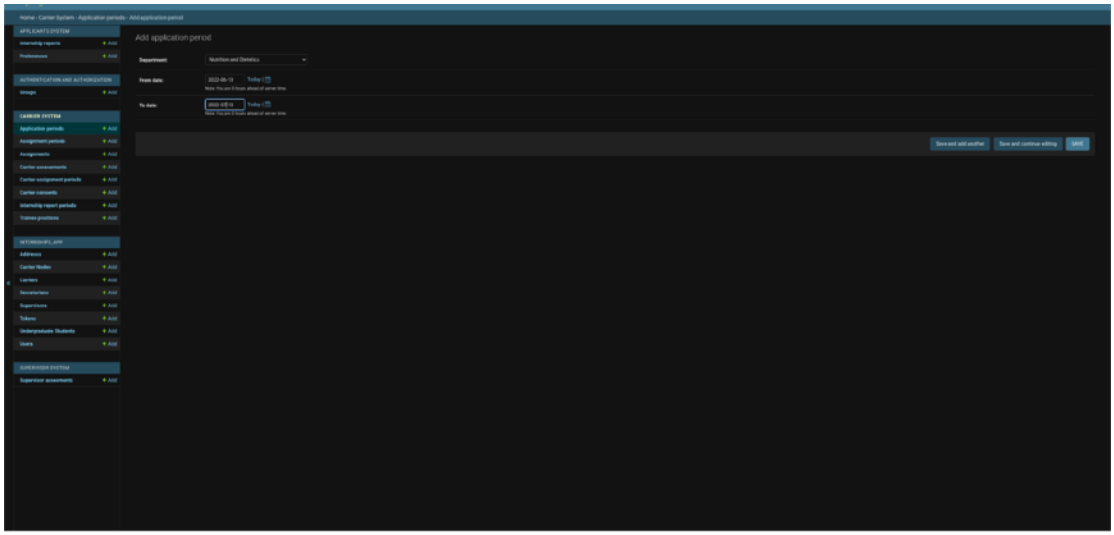


Χρήστες εφαρμογής στο admin

Στο **Applicant System** έχουμε πρόσβαση σε αιτήσεις φοιτητών και στα αρχεία προς αξιολόγηση τους.

Στο **Carrier System** έχουμε τα μοντέλα για όλες τις περιόδους του συστήματος ,αλλά και αντικείμενα όπως θέσεις εργασίας , αναθέσεις και consents (συναινέσεις).

Ενδεικτικά ο Admin είναι υπεύθυνος να ορίσει τις περιόδους του συστήματος όποτε αυτό κριθεί απαραίτητο.



Ορισμός περιόδων Π.Α. συστήματος

Ωστόσο , μεταξύ των χρονικών διαστημάτων δε θα πρέπει να υπάρχουν επικαλύψεις και η σειρά με την οποία θα πρέπει να διεξάγονται οι φάσεις είναι οι εξής:

1.Carrier Assignment Period

2.Application Period

3.Assignment Period

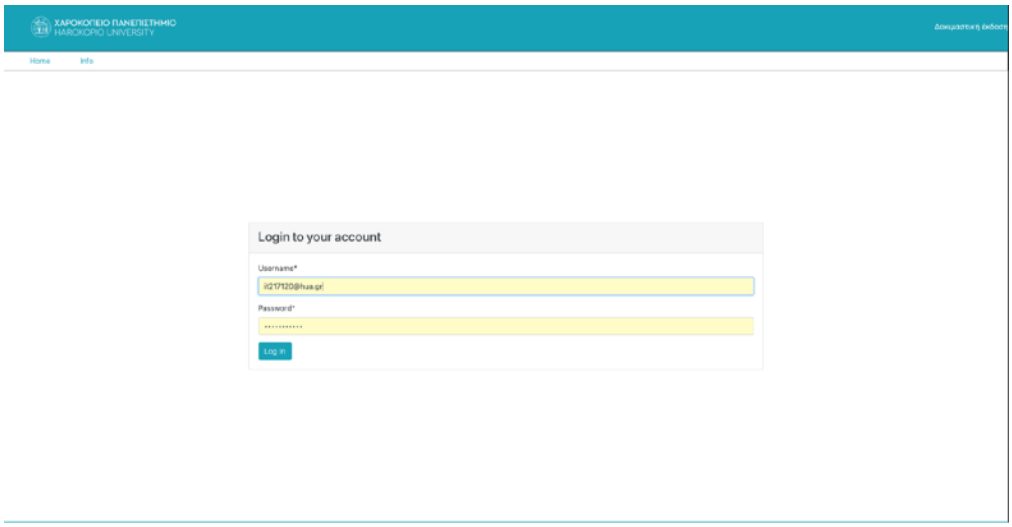
4.Internship Report Period.

Για την αλλαγή κάθε φάσης θα πρέπει η γραμματεία να βρίσκεται σε συνεννόηση με το διαχειριστή του συστήματος , ώστε να μην υπάρξουν λάθη κατά τη μετάβαση σε επόμενες φάσεις του συστήματος.

Στο **Supervisor System** , μπορούμε να δούμε το σύνολο των αξιολογήσεων όλων των επιβλέποντων καθηγητών του συστήματος.

3.3 Σενάρια χρήσης

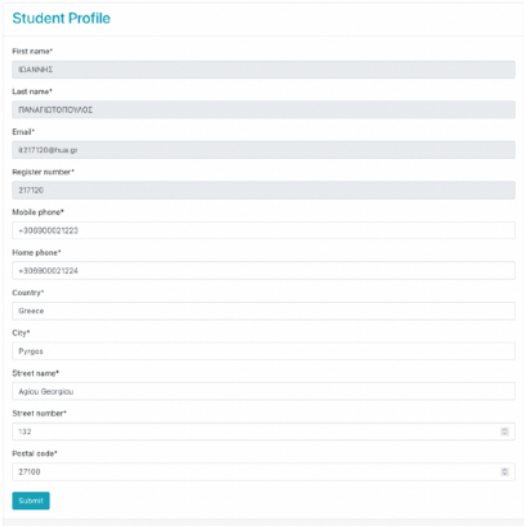
1. Login χρήστη:



Σύνδεση προπτυχιακού φοιτητή

Ο προπτυχιακός φοιτητής συνδέεται με τα στοιχεία του για πρώτη φορά στο σύστημα και αυθεντικοποιείται μέσω της εφαρμογής που επικοινωνεί με τον LDAP server του πανεπιστημίου.

2. Εγγραφή Προπτυχιακού Φοιτητή

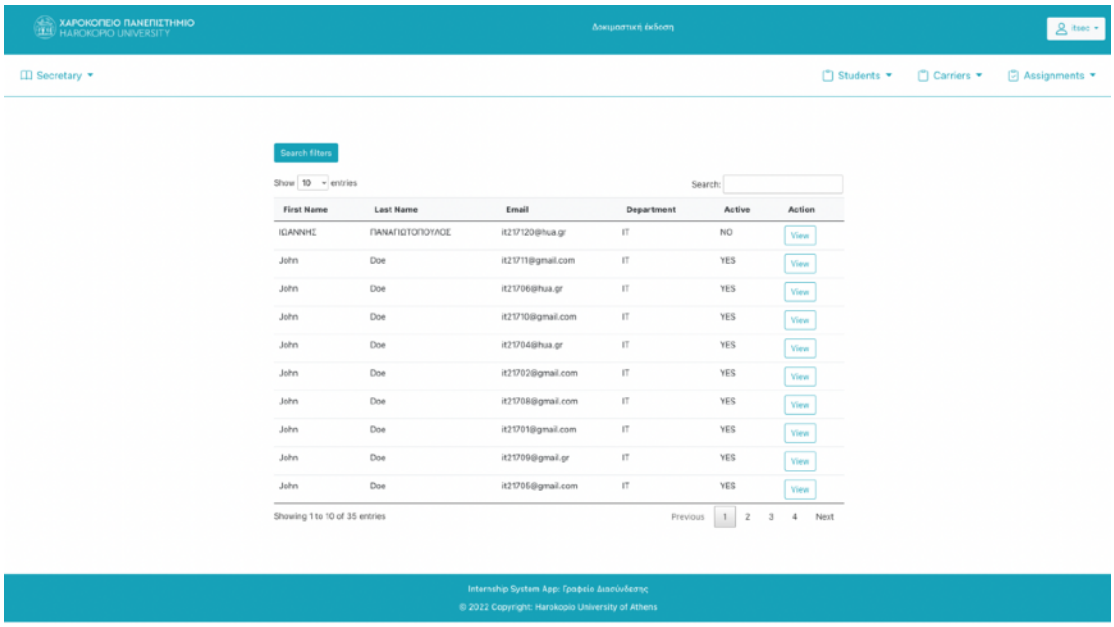


Εγγραφή προπτυχιακού φοιτητή

Εφόσον η σύνδεση του προπτυχιακού φοιτητή ήταν επιτυχής , ο φοιτητής ανακατευθύνεται σε μια σελίδα για να συμπληρώσει τα επιπλέον στοιχεία επικοινωνίας και κατοικίας του.

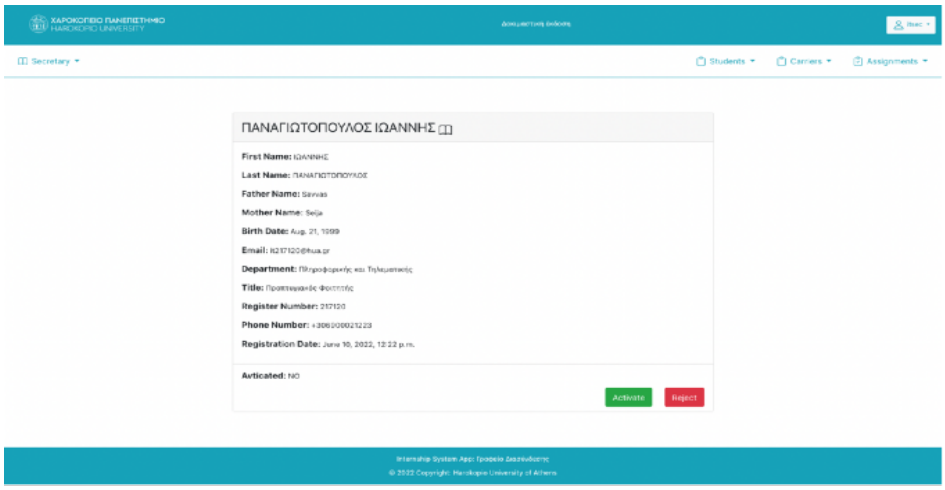
Όταν ολοκληρώσει τη συμπλήρωση των στοιχείων αναμένει την αποδοχή της εγγραφής του από τη γραμματεία.

3. Αποδοχή Φοιτητή από τη Γραμματεία



Λίστα φοιτητών γραμματείας

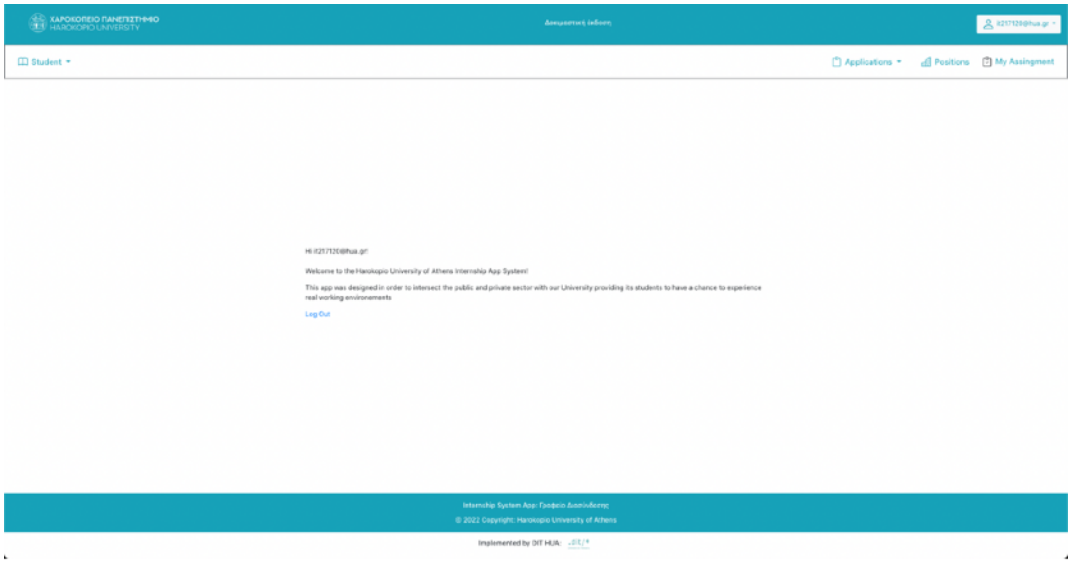
Ο υπάλληλος της γραμματείας βλέπει τη λίστα από τους φοιτητές που έχουν κάνει εγγραφή στο σύστημα. Ωστόσο στις πρώτες εγγραφές συνήθως εμφανίζονται οι εγγραφές των φοιτητών που δεν έχουν γίνει αποδεκτές από το σύστημα(αν υπάρχουν).
Σε αυτό το σημείο ο υπάλληλος της γραμματείας μπορεί πατήσει το κουμπί “View” , για να λάβει μια προεπισκόπηση αυτού του χρήστη.



Αποδοχή φοιτητή στο σύστημα 1

Βλέπουμε τη προεπισκόπηση του χρήστη. Η γραμματεία ελέγχει αν αυτός ο χρήστης είναι έγκυρος και πληροί τα κριτήρια συμμετοχής στο πρόγραμμα και τον αποδέχεται πατώντας το κουμπί “Activate”
Σε περίπτωση που πατήσει “Reject” ο φοιτητής διαγράφεται από τη Β.Δ.

4.Είσοδος φοιτητή στο σύστημα

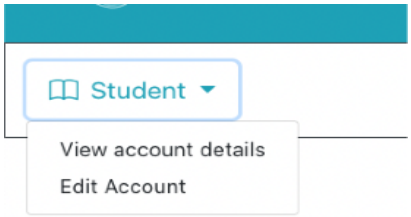


Αρχική προφίλ φοιτητή

Στην ενότητα Student φοιτητής έχει τις εξής επιλογές:

1.Να δει τα προσωπικά του στοιχεία(View account details).

2.Να επεξεργαστεί τα προσωπικά του στοιχεία(Edit account).



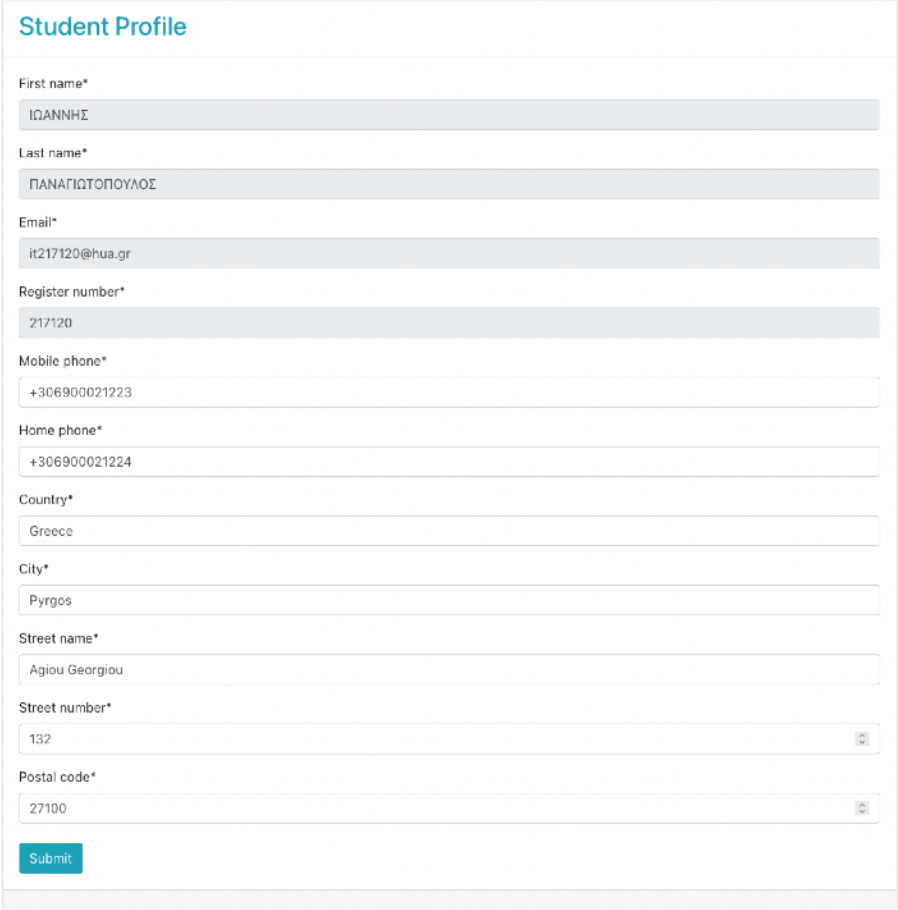
Επιλογές προφίλ φοιτητή

Οι επιλογές φαίνονται στο παραπάνω στιγμιότυπο.

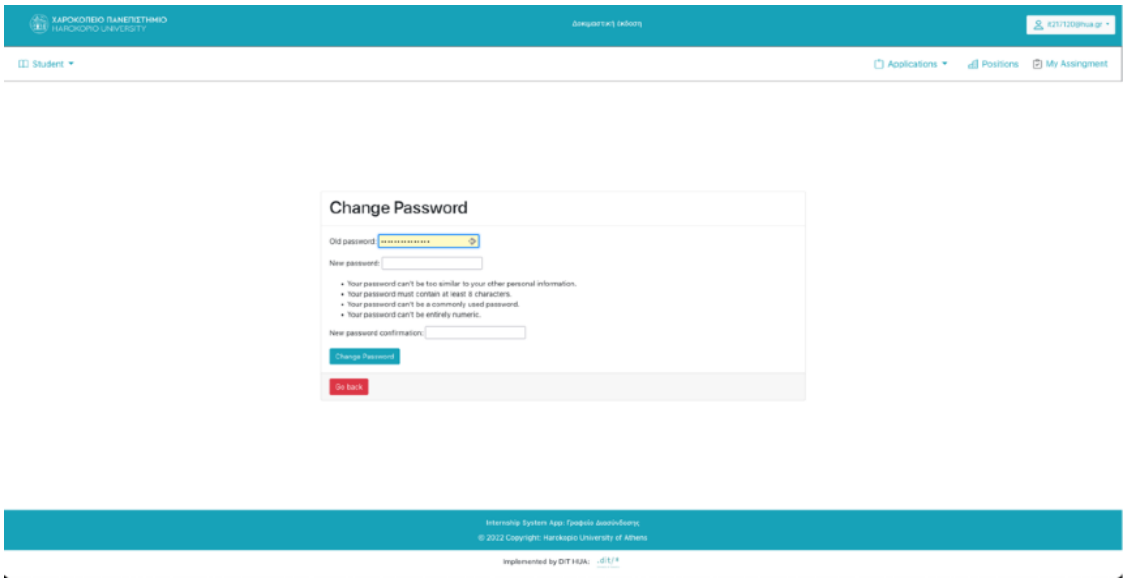
5.Αλλαγή στοιχείων χρήστη

Ο χρήστης μπορεί να αλλάξει τα στοιχεία επικοινωνίας του και τη διεύθυνση. Ωστόσο για να αλλαχθούν στοιχεία όπως το όνομα του θα πρέπει να επικοινωνήσει με τη γραμματεία του τμήματος που θα απευθυνθεί στο διαχειριστή του συστήματος.

Αλλαγή στοιχείων φοιτητή



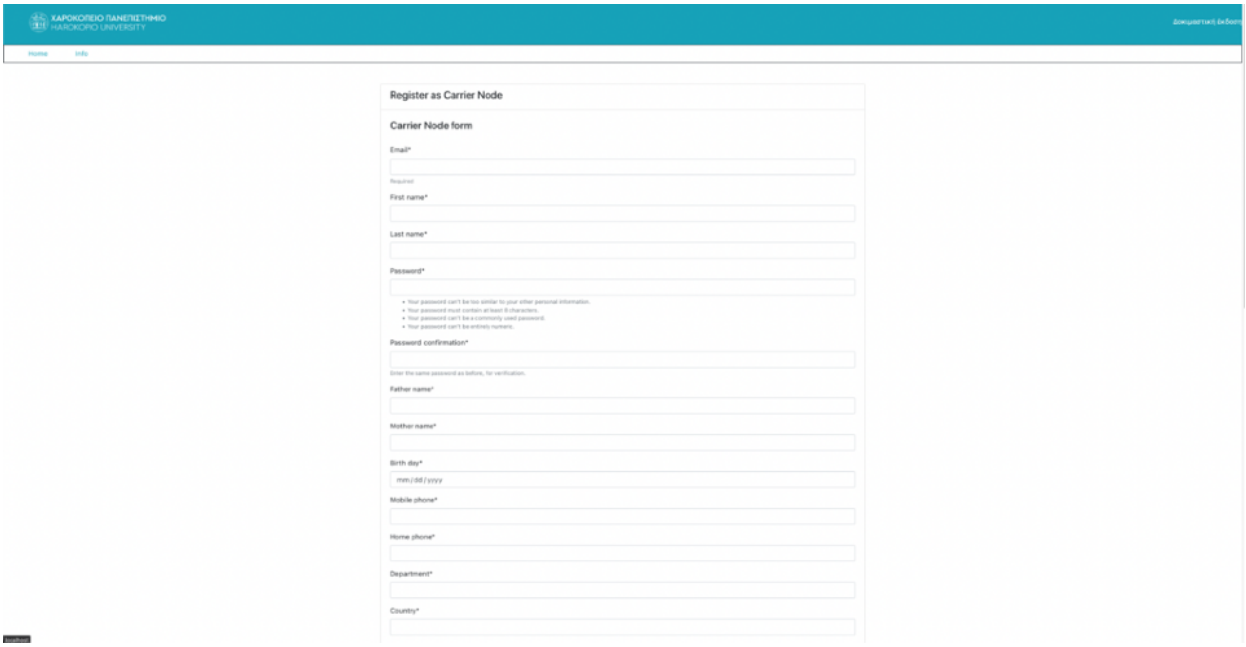
6.Φόρμα αλλαγής κωδικού



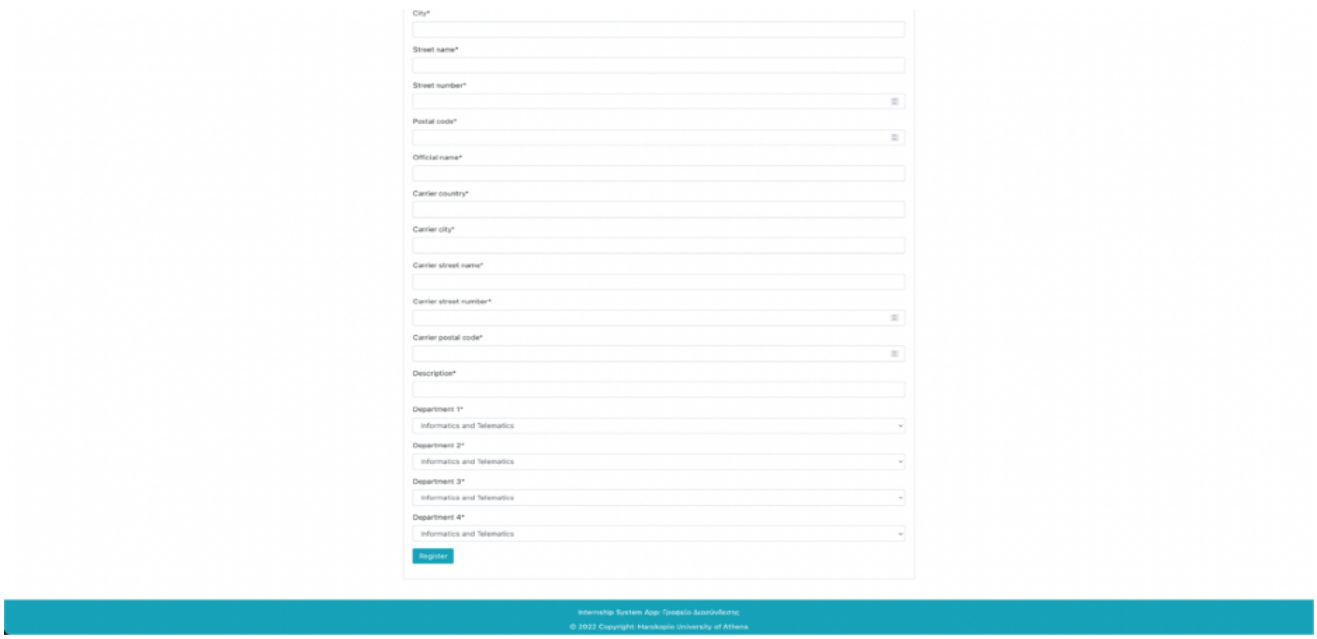
Φόρμα αλλαγής κωδικού

Οι χρήστες εκτός των φοιτητών και των καθηγητών που το username τους είναι το ακαδημαϊκό email , έχουν τη δυνατότητα να αλλάζουν το κωδικό πρόσβασης μέσω αυτής της φόρμας.

7.Εγγραφή φορέα.



Εγγραφή φορέα



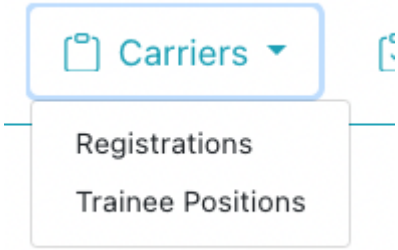
Εγγραφή φορέα 2

Το Σ.Ε.Φ. εγγράφεται στο σύστημα συμπληρώνοντας τα προσωπικά στοιχεία και τα στοιχεία του φορέα. Εφόσον ολοκληρώσει τη διαδικασία επιτυχώς , περιμένει την έγκριση της συμμετοχής του από τη γραμματεία.

8.Έγκριση συμμετοχής φορέα.

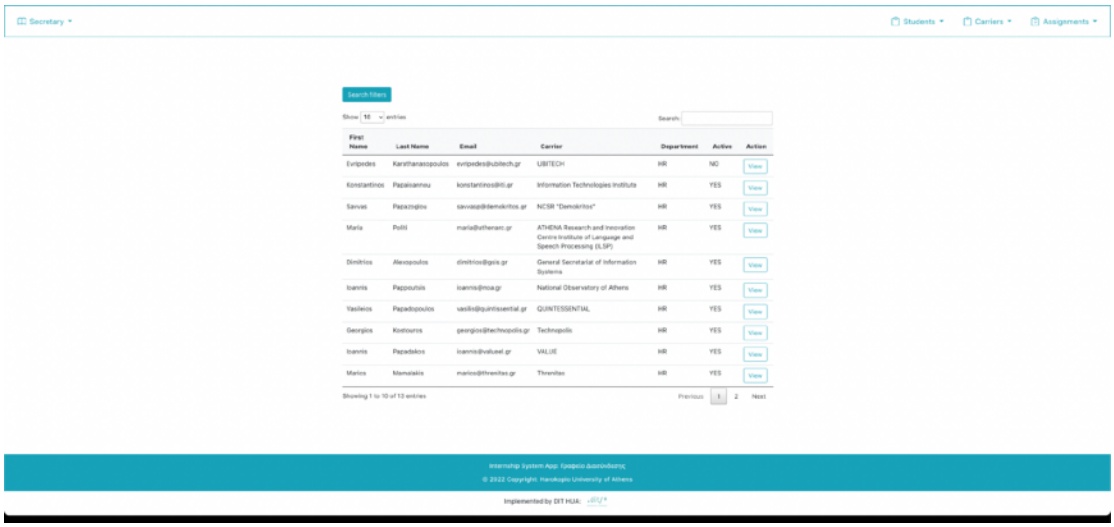
Στο βασικό μενού της γραμματείας , που βρίσκεται στο πάνω μέρος της σελίδας κάτω από το πράσινο πλαίσιο εντοπίζουμε το στοιχείο Carriers(Φορείς) με τις εξής δύο επιλογές:

1.Registrations (Εγγραφές)
2.Trainee Positions (Θέσεις Π.Α)



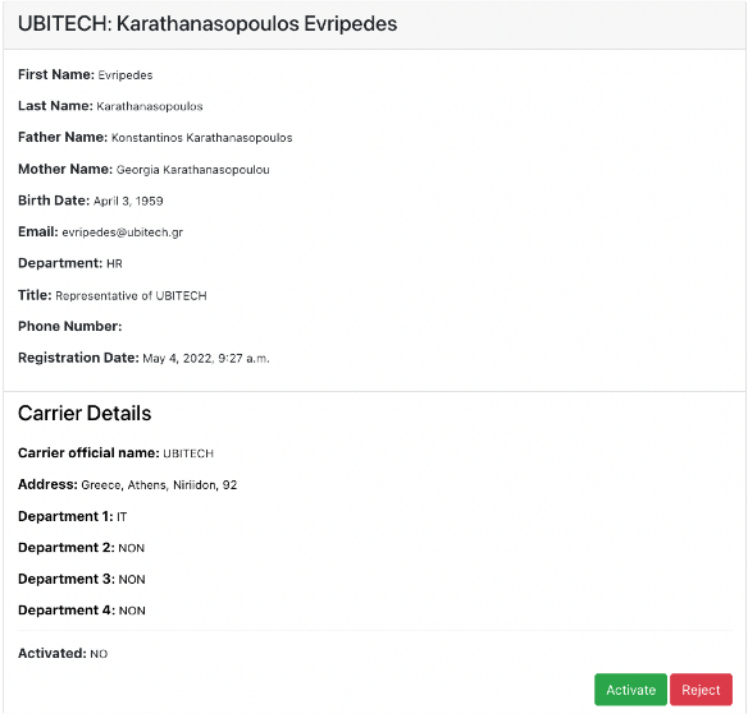
Επιλογές προφίλ ΣΕΦ

Εφόσον επιλεχθεί το "Registrations" , τότε μπορούμε να δούμε ένα πίνακα με το σύνολο των φορέων που έχουν ολοκληρώσει ή αιτούνται εγγραφή στο σύστημα.



Λίστα φορέων γραμματείας

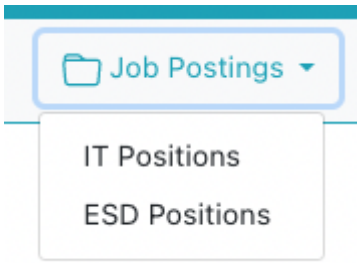
Στη πρώτη σειρά μπορούμε να δούμε ένα Σ.Ε.Φ. με το φορέα του , που δεν έχει εγκριθεί ακόμα από τον υπάλληλο της γραμματείας.
Πατώντας το κουμπί "View" , μπορούμε να προβούμε σε προεπισκόπηση των στοιχείων της εγγραφής του στο σύστημα.



Προεπισκόπηση φορέα

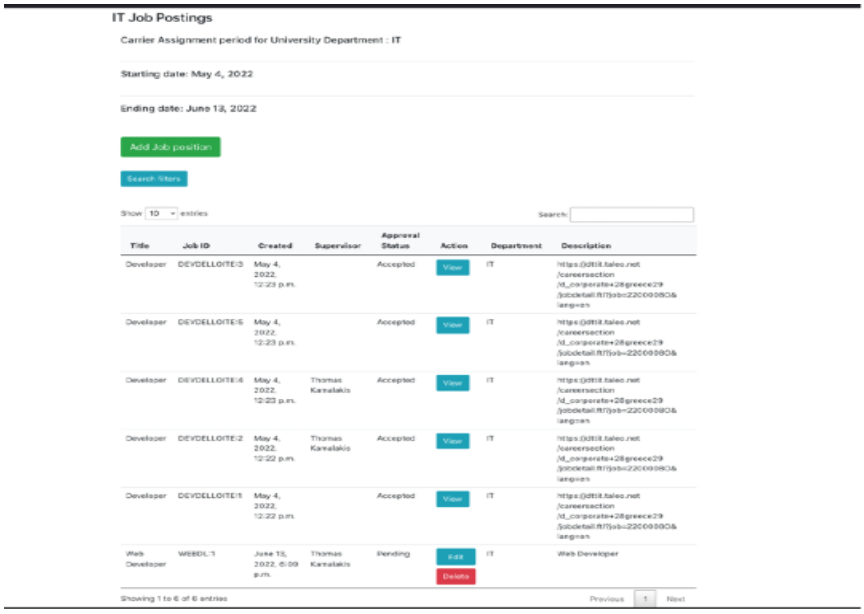
9.Δημιουργία θέσης πρακτικής άσκησης.

Το Σ.Ε.Φ. μέσω του δικού του προφίλ , έχει τη δυνατότητα να εισάγει θέσεις πρακτικής άσκησης για κάθε τμήμα του πανεπιστημίου στο οποίο ο φορέας του απευθύνεται.
Η επιλογή αυτή βρίσκεται στο κουμπί “Job Postings” στο προφίλ του φορέα , όπου κατά το πάτημα του μπορεί να δει τα αντίστοιχα διαθέσιμα τμήματα.
Οι θέσεις πρακτικής άσκησης έχουν ένα μοναδικό κωδικό για το είδος της θέσης (job_code) και ένα κωδικό για τον υπ’ αριθμό ζητούμενο ασκούμενο για αυτή τη θέση.
Έστω ότι έχουμε μια θέση με τίτλο Web Developer για μια εταιρεία X , ο κωδικός (job_code) που μπορεί να μπει ενδεικτικά για το είδος της θέσης είναι XWEBDEV και η θέση(job_id) για το πρώτο εργαζόμενο 01 , και για το δεύτερο 02.
Για το job_id έχουν προβλεφθεί στο σύνολο 10 θέσεις , οπότε το job_id μπορεί να λάβει τιμές από 1-10.



Επιλογές τμημάτων

Ενημέρωση στοιχείων εργασίας



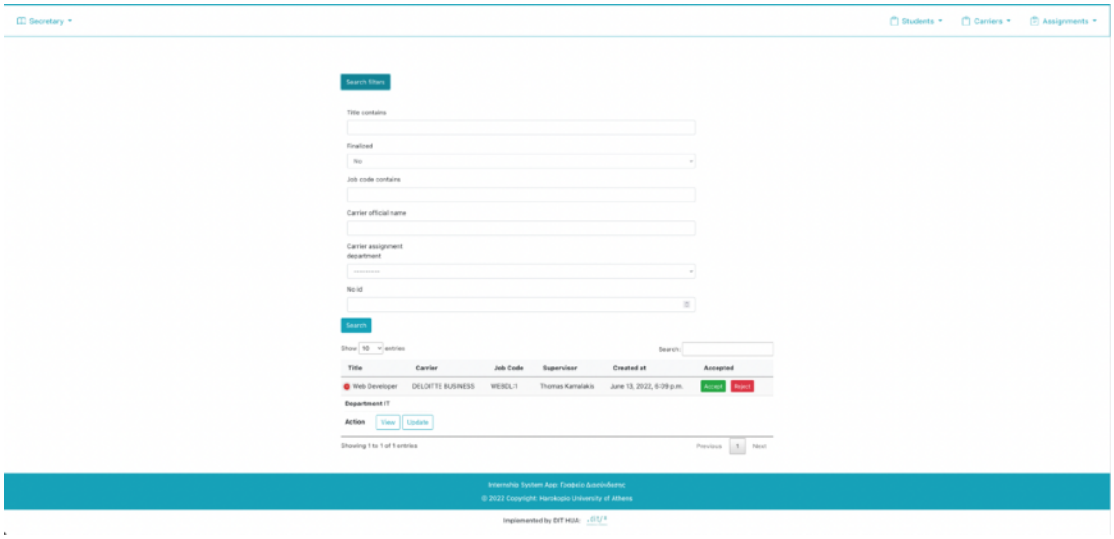
Λίστα θέσεων εργασίας ΣΕΦ

Παραπάνω βλέπουμε το στιγμιότυπο στο οποίο το Σ.Ε.Φ βλέπει τις θέσεις που έχει υποβάλλει στο σύστημα με τη κατάσταση του αιτήματος τους. Σε περίπτωση που δεν έχει ληφθεί τελική απάντηση από τη γραμματεία , μπορεί είτε να διαγράψει ή να επεξεργαστεί την αίτηση του.

10. Έγκριση θέσης Π.Α. από τη γραμματεία.

Εφόσον τα Σ.Ε.Φ. εισάγουν τις θέσεις Π.Α. που ενδιαφέρουν το φορέα τους , οι γραμματείες των αντίστοιχων τμημάτων του πανεπιστημίου λαμβάνουν το αίτημα τους και το επικυρώνουν ή ακυρώνουν ανάλογα με τις αποφάσεις που λαμβάνονται από τις αρμόδιες επιτροπές για τη Π.Α. .

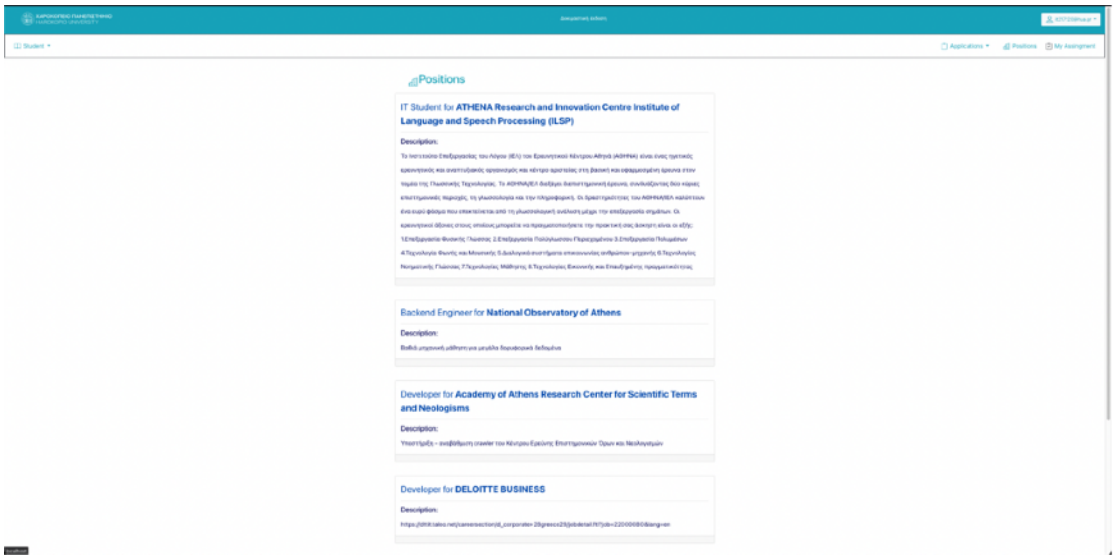
Στο προφίλ της γραμματείας στο πάνω δεξί μέρος του μενού που βρίσκεται κάτω από το πράσινο πλαίσιο εντοπίζουμε την επιλογή “Carriers” και εφόσον πατηθεί, οδηγούμαστε την επιλογή “Trainee Positions”.Εφόσον οδηγηθούμε στο “Trainee Positions”, ανακατευθυνόμαστε σε μια σελίδα όπου εντοπίζουμε το σύνολο των θέσεων πρακτικής άσκησης που έχουν εγκριθεί ή αναμένεται να ληφθεί η τελική απόφαση της επί της έγκρισης τους ή απόρριψης τους.
Σε περίπτωση απόρριψης οι θέσεις διαγράφονται από το σύστημα και τη Β.Δ. .



Έγκριση Π.Α. γραμματείας 1

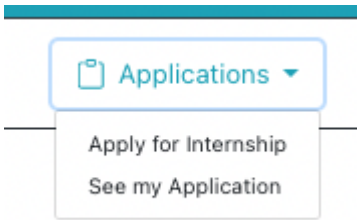
11. Αίτηση ενδιαφέροντος για Π.Α.

Κατά τη περίοδο **ApplicationPeriod** οι προπτυχιακοί φοιτητές συμπληρώνουν την αίτηση ενδιαφέροντος για τη Π.Α. κατά το τρέχον διάστημα. Έχουν τη δυνατότητα να δουν τις διαθέσιμες θέσεις Π.Α. από όλους τους φορείς που απευθύνονται στο τμήμα τους. Στη συνέχεια συμπληρώνουν την αίτηση η οποία περιέχει από 1 έως 5 επιλογές για τη Π.Α. με αντίστοιχη σειρά προτεραιότητας.



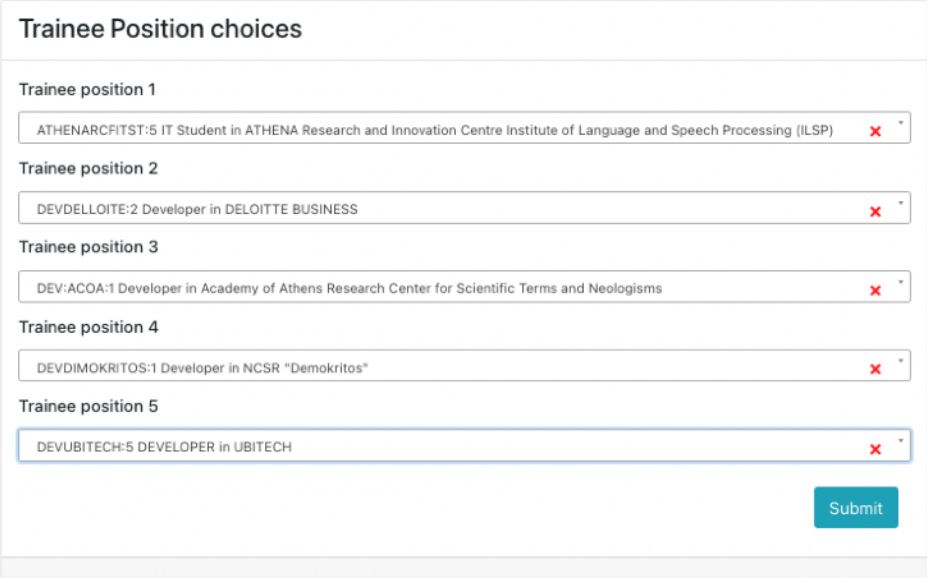
Θέσεις εργασίας φοιτητή

Στο προφίλ του φοιτητή κάτω στο πάνω μέρος δεξιά κάτω από το πράσινο πλαίσιο εντοπίζουμε το κουμπί “Applications”, στο οποίο έχουμε την επιλογή “Apply for internship”. Πατάμε την επιλογή αυτή και έπειτα μας εμφανίζεται η φόρμα με τις προτιμήσεις για την Π.Α.



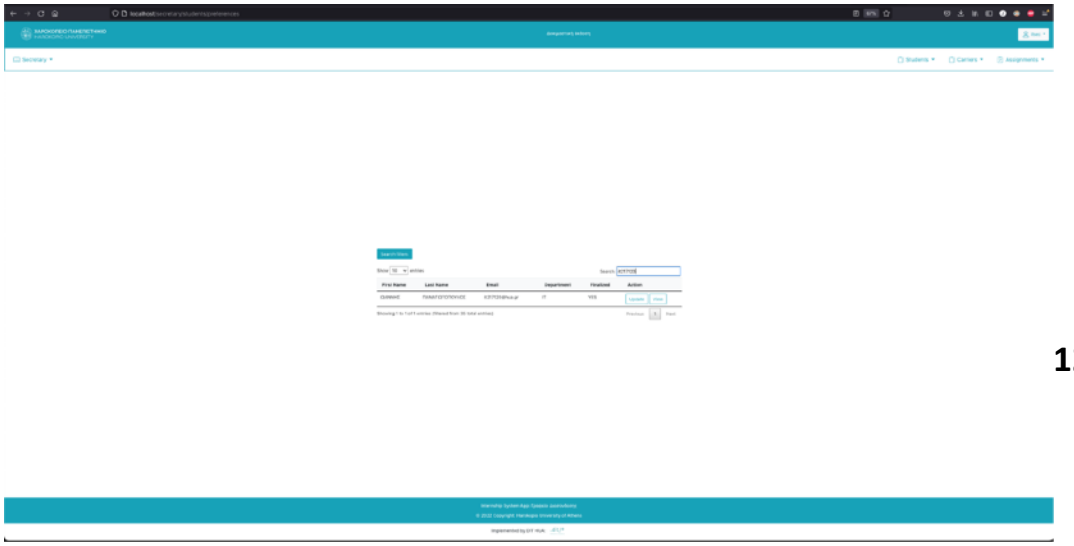
Επιλογές αίτησης φοιτητή

Εφόσον λήξει η περίοδος ενδιαφέροντος δηλώσεων Π.Α . Οι γραμματείες των τμημάτων ελέγχουν τις αιτήσεις των φοιτητών και τις οριστικοποιούν.



Επιλογές θέσεων εργασίας φοιτητή

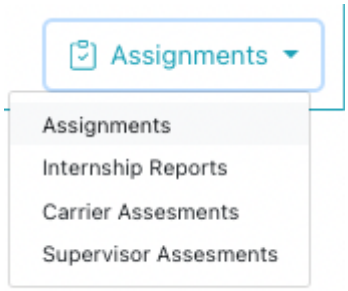
Λίστα αιτήσεων φοιτητών



12.

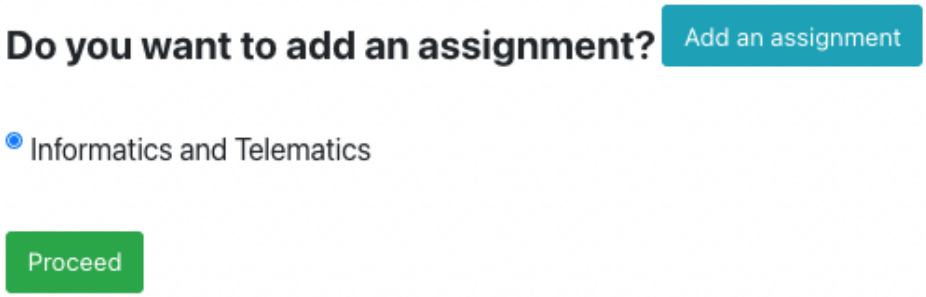
Δημιουργία προτάσεων αναθέσεων Π.Α.

Εφόσον το ApplicationPeriod τελειώσει και οι αιτήσεις των φοιτητών ελεγχθούν πλήρως, τότε οι γραμματείες των τμημάτων δημιουργούν κάποιες αναθέσεις για τους αιτούντες φοιτητές λαμβάνοντας υπόψη τις προτιμήσεις τους.
Στο προφίλ του υπαλλήλου της γραμματείας πηγαίνουμε στο Assignmens->Assignments.



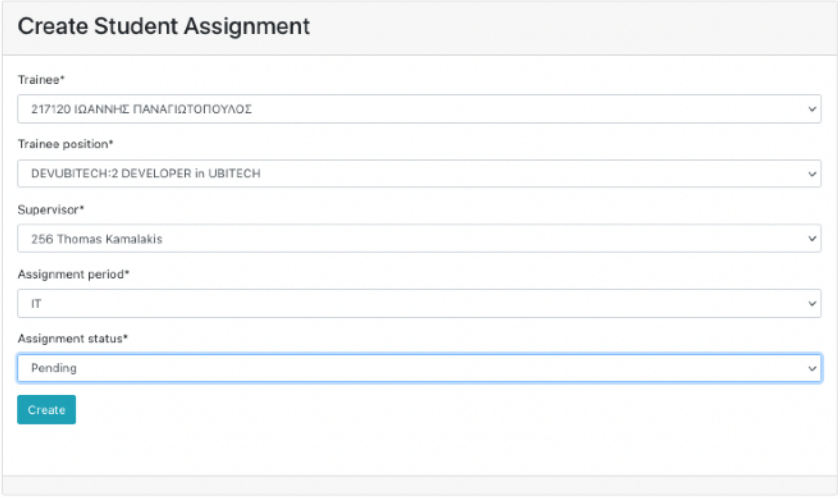
Επιλογή θέσεων γραμματείας

και έπειτα επιλέγουμε το τμήμα για το οποίο θέλουμε να δημιουργήσουμε μια υποψήφια ανάθεση.



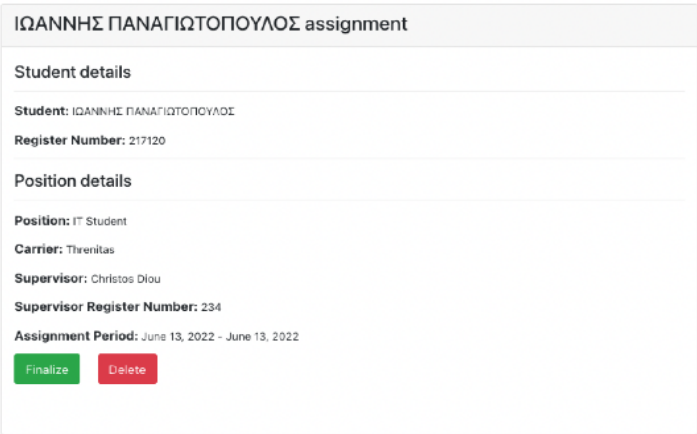
Επιλογές τμημάτων για ανάθεση

Εν συνεχεία , οδηγούμαστε στη φόρμα για τη συμπλήρωση των στοιχείων της ανάθεσης:

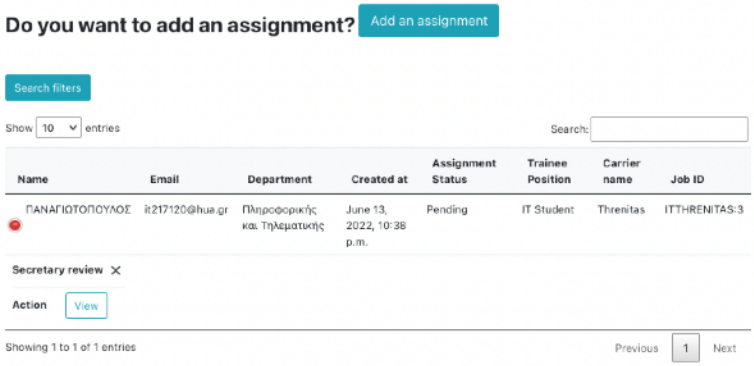


Δημιουργία υποψήφιας ανάθεσης

Όπως βλέπουμε παρακάτω η ανάθεση είναι πρόχειρη(draft), οπότε για να οριστικοποιήσουμε την υποψήφια ανάθεση θα πρέπει να την κάνουμε finalize.



Οριστικόποιηση υποψήφιας ανάθεσης

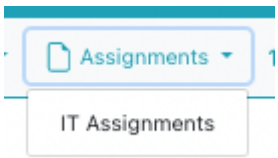


Επισκόπηση αναθέσεων

13. Οριστικοποίηση ανάθεσης Π.Α.

Τα Σ.Ε.Φ. λαμβάνουν τις υποψήφιες αναθέσεις και αποφασίζουν για το αν συναινούν ή όχι σε μια ανάθεση που πρόκειται για ένα φοιτητή του τμήματος στον οποίο απευθύνεται η εν λόγω θέση εργασίας της ανάθεσης.

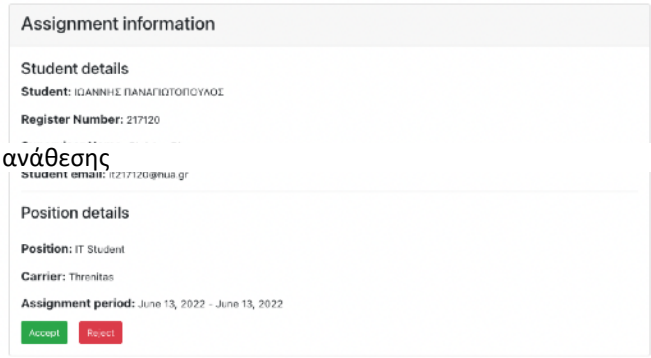
Τα Σ.Ε.Φ. επιλέγουν το στοιχείο Assignments στο μενού κάτω από το πράσινο πλαίσιο που βρίσκεται στο πάνω μέρος της σελίδας τους.



Επιλογές αναθέσεων

Μετά την επιλογή προβολής των υποψήφιων αναθέσεων για ένα τμήμα του πανεπιστημίου. Λαμβάνουν μια λίστα με υποψήφιες αναθέσεις που αναφέρονται στις θέσεις πρακτικής άσκησης που έχουν αναρτήσει στο σύστημα. Σε αυτό το σημείο μπορούν να δουν τα στοιχεία του υποψήφιου ασκούμενου καθώς και τα στοιχεία επικοινωνίας του

Οι φορείς βλέπουν την καρτέλα του υποψηφίου , επικοινωνούν μαζί του εκτός πλατφόρμας και έπειτα αποφασίζουν για την πρόσληψη του.

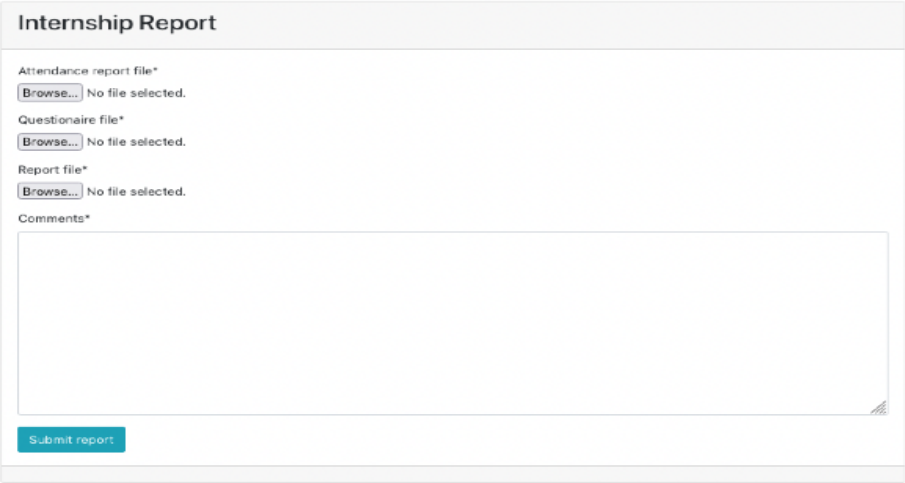


Προβολή ανάθεσης

Αποδοχή ανάθεσης ΣΕΦ

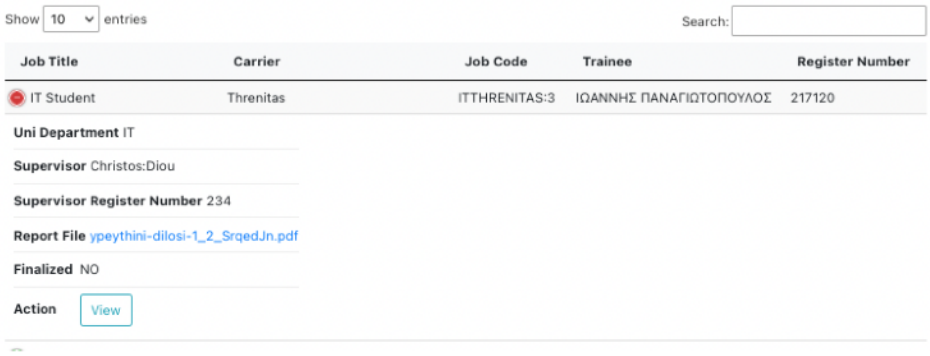
14. Βαθμολόγηση φοιτητή στη Π.Α.

Όταν ο ασκούμενος ολοκληρώσει τη Π.Α. του , τότε έχει τη δυνατότητα να καταθέσει το αρχείο αξιολόγησης του. Το αρχείο αυτό περιλαμβάνει το παρουσιολόγιο το οποίο χρειάζεται η γραμματεία , το ερωτηματολόγιο αξιολόγησης του φορέα από το φοιτητή , αλλά και την έκθεση πεπραγμένων που είναι το κύριο αντικείμενο για την αξιολόγηση του φοιτητή.



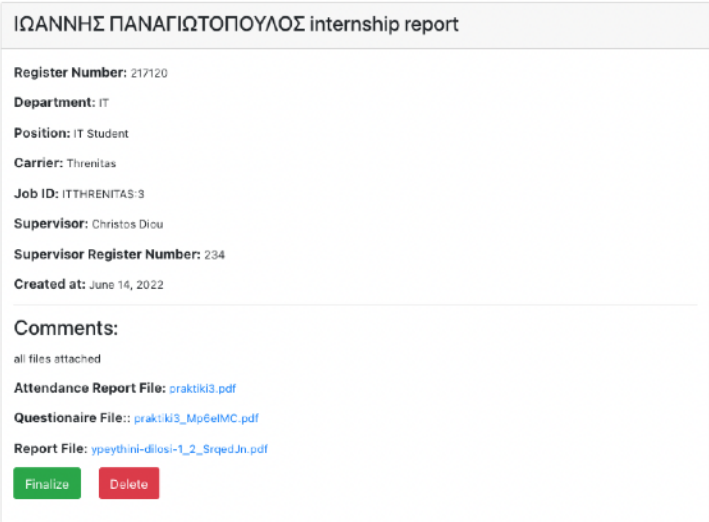
Αρχείο αξιολόγησης φοιτητή

Εφόσον ο φοιτητής ανεβάσει το αρχείο αξιολόγησης του , η γραμματεία αναλαμβάνει να ελέγξει το αρχείο και να εγκρίνει το ανέβασμα του ή να το απορρίψει. Σε περίπτωση απόρριψης , η γραμματεία αναλαμβάνει να επικοινωνήσει με το φοιτητή εκτός πλατφόρμας και να του επισημάνει ποιος είναι ο σωστός τρόπος υποβολής.



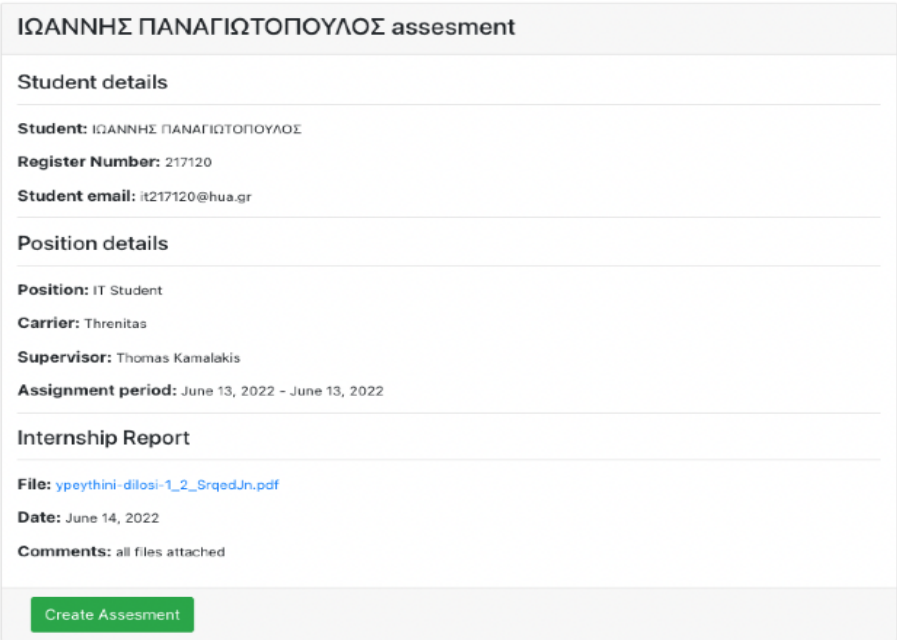
Προβολή αρχείων αξιολόγησης φοιτητών

Εφόσον το αρχείο οριστικοποιηθεί , τότε ενημερώνονται οι φορείς για το ανέβασμα της έκθεσης πεπραγμένων του ασκούμενου και αναλαμβάνουν τη βαθμολόγηση του.



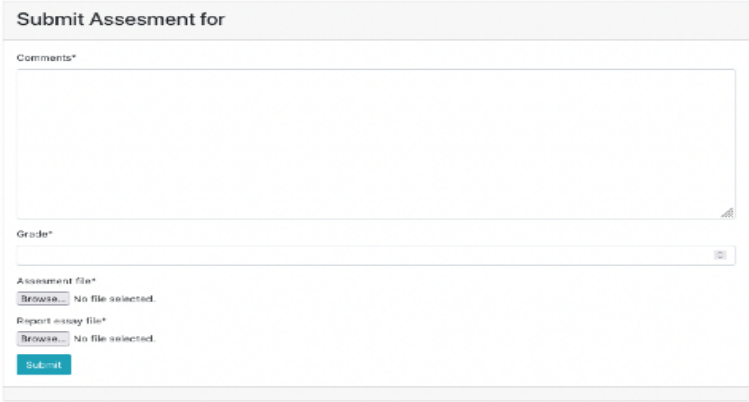
Οριστικοποιήση Α.Α. φοιτητή 1

Τα Σ.Ε.Φ. ενημερώνονται πως ο φοιτητής έχει υποβάλλει το αρχείο αξιολόγησης του και με τη σειρά τους μπορούν να δουν την αναρτημένη έκθεση πεπραγμένων και με βάση αυτή ανεβάζουν ένα αρχείο με τις παρατηρήσεις τους πάνω σε αυτήν , καθώς και ένα ξεχωριστό αρχείο αξιολόγησης του φοιτητή.



Δημιούργια αξιολόγησης ΣΕΦ

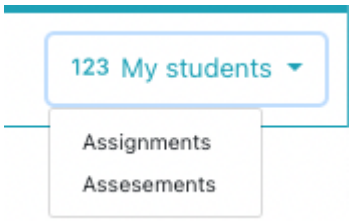
Το Assesement file είναι για την αξιολόγηση του φοιτητή. Το Report essay file είναι για τα σχόλια του φορέα πάνω στην έκθεση πεπραγμένων του ασκούμενου.
Η γραμματεία αναλαμβάνει να ελέγξει το αρχείο αξιολόγησης του φορέα για τον



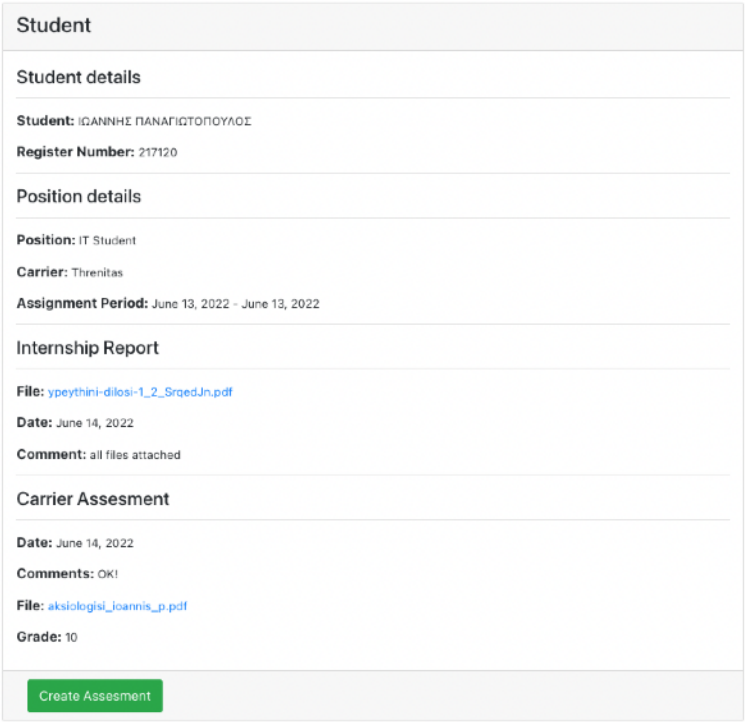
Αξιολόγηση φοιτητή από ΣΕΦ

κάθε ασκούμενο και να το επικυρώσει σε περίπτωση που είναι έγκυρο. Σε περίπτωση που υπάρχουν σφάλματα , επικοινωνεί με το Σ.Ε.Φ. για τις λεπτομέρειες περί της μη έγκυρης μορφής του αρχείου και ζητά την επανάναρτηση των αρχείων.

Εφόσον οριστικοποιηθεί η αξιολόγηση του φορέα , οι καθηγητές πλέον με βάση τα αρχεία του φοιτητή και του φορέα μπορούν να υποβάλλουν το αρχείο με τη τελική βαθμολογία του ασκούμενου φοιτητή.
Για να δουν τις αναθέσεις προς αξιολόγησης επιλέγουν το κουμπί “My Students” και έπειτα την επιλογή “Assignments”.



Επιλογές αξιολογήσεων καθηγητών



Student

Student details

Student: ΙΩΑΝΝΗΣ ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ

Register Number: 217120

Position details

Position: IT Student

Carrier: Threnitas

Assignment Period: June 13, 2022 - June 13, 2022

Internship Report

File: ypeythini-dilosi-1_2_SrqedJn.pdf

Date: June 14, 2022

Comment: all files attached

Carrier Assesment

Date: June 14, 2022

Comments: OK!

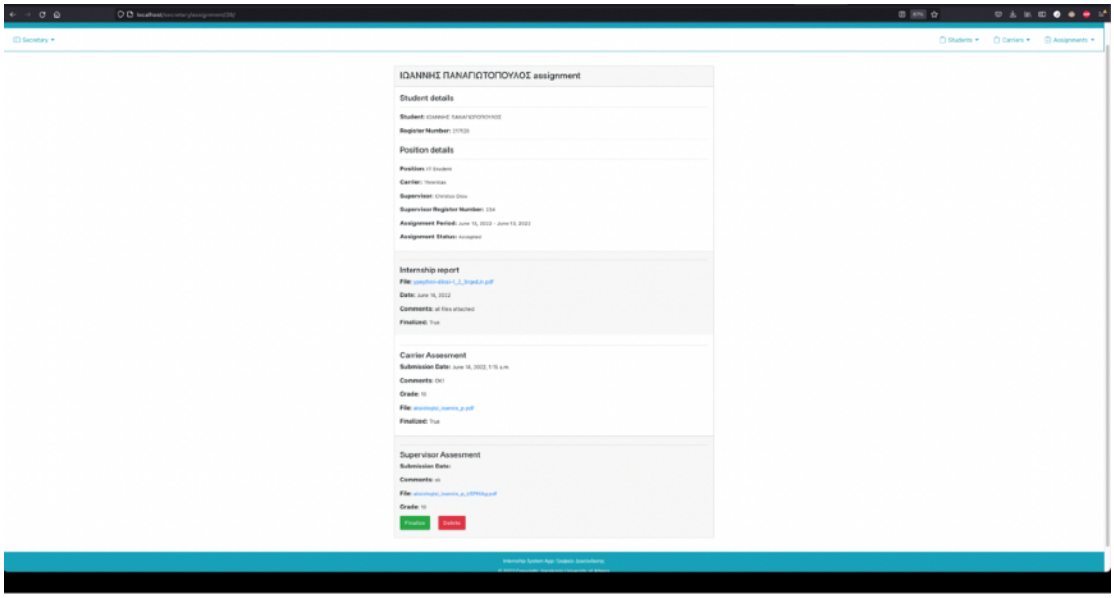
File: aksiologisi_ioannis_p.pdf

Grade: 10

Create Assesment

Συνολική αναφορά Π.Α. καθηγητή 1

Όπως βλέπετε , ο καθηγητής έχει μια συνολική εικόνα για τη πρακτική του εκάστοτε φοιτητή με πρόσβαση στα αρχεία αξιολόγησης του.



Αποδοχή αξιολόγησης καθηγητή

Εφόσον συμπληρώσει τη τελική βαθμολογία , η γραμματεία μπορεί να οριστικοποιήσει το συνολικό αρχείο αξιολόγησης και να ανεβάσει σε δεύτερο χρόνο τους τελικούς βαθμούς στο φοιτητολόγιο.

ΚΕΦ.4.ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Στη παρούσα εργασία , βασικός στόχος ήταν να αναπτύξουμε ένα Ελάχιστο Βιώσιμο Προϊόν (MVP) για ένα σύστημα παρακολούθησης Π.Α. .Γενικά δόθηκε έμφαση στην απλότητα των διαδικασιών που εκτυλίσσονται εντός του συστήματος με σκοπό να γίνει πιο προσιτό προς τους τελικούς χρήστες αλλά και πιο εύκολα διαχειρίσιμο από τους υπεύθυνους της Π.Α.(Γραμματεία , Admin). Η δυσκολία στην υλοποίηση αυτού του συστήματος έγκειται στο γεγονός πως υπάρχουν αρκετές φάσεις στο σύστημα και οι διαδικασίες έπρεπε να είναι αυστηρά περιορισμένες πάνω στα διαστήματα αυτά και οι δικαιοδοσίες του κάθε χρήστη θα έπρεπε να είναι σαφώς ορισμένες εκ των προτέρων λαμβάνοντας υπόψη τις κανονιστικές απαιτήσεις για τη διεξαγωγή της Π.Α. .

Ωστόσο μπορούμε να επισημάνουμε κάποιες πτυχές αυτής της εργασίας οι οποίες επιδέχονται βελτιώσεις ή νέες επεκτάσεις , με σκοπό να αναπτυχθούν νέες λειτουργίες και το σύστημα να γίνει μελλοντικά βιώσιμο(future-proof):

Integration με το φοιτητολόγιο:

Θα μπορούσαμε να υλοποιήσουμε μια υπηρεσία εντός του πανεπιστημίου η οποία θα διαθέτει ένα API για την παροχή των δεδομένων για τη κατάσταση φοίτησης κάθε φοιτητή. Αυτό θα ήταν ιδιαίτερα χρήσιμο , στη περίπτωση που θα αυτοματοποιούσαμε τις εγκρίσεις των φοιτητών που αιτούνται εγγραφή στο σύστημα , ελέγχοντας με αυτόματο τρόπο αν πληρούν τα κριτήρια για τη συμμετοχή στη Π.Α.

Statistics Board:

Τα δεδομένα που αντλούνται από αυτό το σύστημα είναι πολλά και ζωτικής σημασίας για την ακαδημαϊκή κοινότητα. Θα μπορούσαμε να φτιάξουμε ένα περιβάλλον στο οποίο θα συγκεντρώνουμε τα στατιστικά για τις προτιμήσεις των φοιτητών(κλάδους πληροφορικής , φορείς, ρόλους εργασίας κ.τ.λ.π.) και να τα οπτικοποιήσουμε εντός της πλατφόρμας για τις γραμματείες των τμημάτων. Επίσης , θα μπορούσαμε να παράξουμε διάφορα csv(comma separated values) αρχεία με τη τρέχουσα κατάσταση της Π.Α. με σκοπό να χρησιμοποιηθούν μελλοντικά για ερευνητικούς σκοπούς(ανάλυση δεδομένων) με σκοπό να δούμε τις σημερινές τάσεις για την επιλογή επαγγελμάτων που έχουν οι φοιτητές του τμήματος.

Σπάσιμο της εφαρμογής σε microservices:

Σε περίπτωση που θέλουμε να αναπτύξουμε την εφαρμογή με αρκετές και σύνθετες λειτουργίες , κρίνουμε σκόπιμο το σπάσιμο της εφαρμογής σε μικρότερα τμήματα με την ανάπτυξη διαφόρων REST APIs που το καθένα θα εξυπηρετεί ένα συγκεκριμένο σκοπό.Η εφαρμογή των χρηστών θα μπορούσε να υλοποιηθεί σε κάποιο frontend framework(React, Angular, Vue) , για το στήσιμο των APIs θα χρησιμοποιούσαμε το Django REST Framework ή το FastAPI και για σύστημα διαχείρισης χρηστών το Keycloak με σκοπό να ορίζουμε δικαιοδοσίες και ρόλους στους χρήστες. Μια τέτοια αρχιτεκτονική προσέγγιση κρίνεται ιδανική όταν ένα λογισμικό είναι αρκετά σύνθετο και ταχεία αναπτυσσόμενο σε μια σχετική μεγάλη ομάδα προγραμματιστών , αλλά η συνδεσμολογία θα πρέπει να είναι σαφής εκ των προτέρων ώστε να εξυπηρετεί τους σκοπούς αυτού του συστήματος.

Προσθήκη ενημερώσεων στο σύστημα και ανάπτυξη chatting εφαρμογής εντός του συστήματος.

Στο σύστημα αυτό θα μπορούσαμε να έχουμε ενημερώσεις σχετικά με την εξέλιξη των πρακτικών ασκήσεων για όλα τα εμπλεκόμενα μέλη.
Επιπροσθέτως ένα ενσωματωμένο chatting σύστημα μεταξύ της γραμματείας και των τελικών χρηστών θα ήταν ιδανικό ώστε να μπορούν τα μέλη να επικοινωνούν για θέματα που αφορούν αποκλειστικά τις πρακτικές ασκήσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. History of Python
2. What is Python?Executive Summary
3. Django History
4. Django Introduction
5. Views in Django
6. Introduction to Django ORM
7. django-crispy-forms
8. WEB II Προηγμένος σχεδιασμός
9. PostgreSQL: About
10. Τι είναι το Nginx;
11. Τι είναι το Docker και πώς το εγκαθιστούμε;