



ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

**ΣΧΟΛΗ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη διαδικτυακής εφαρμογής για την online
δημιουργία μουσικών κοινωνιών**

Γεώργιος Π. Πανόπουλος

**Επιβλέποντες : Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής
Δημήτριος Τσουμάκος, Αναπληρωτής Καθηγητής**

ΚΕΡΚΥΡΑ

ΜΑΙΟΣ 2019

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανάπτυξη διαδικτυακής εφαρμογής για την online δημιουργία
μουσικών κοινωνιών**

Γεώργιος Π. Πανόπουλος

ΑΜ: Π2014062

Επιβλέποντες : Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής

Δημήτριος Τσουμάκος, Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία υλοποιήθηκε μια διαδικτυακή εφαρμογή και ένα website μέσω του framework django της python. Στόχος του συγκεκριμένου web site είναι να βοηθά τους μουσικούς κάθε είδους μουσικής προτίμησης και μουσικής ιδιότητας να βρίσκουν ο ένας τον άλλον εύκολα και άμεσα. Στην εισαγωγή γίνεται μια γενική περιγραφή της σημερινής καθημερινότητας σχετικά με το διαδίκτυο και την υπερπληροφόρηση, εξηγούνται οι λόγοι για τους οποίους επιλέχθηκε το παραπάνω framework και πώς μπορεί μια διαδικτυακή εφαρμογή φτιαγμένη στο django να βοηθήσει μουσικούς στην πραγματικότητα της υπερπληροφόρησης. Στο πρώτο κεφάλαιο περιγράφονται τα frameworks, και οι κατηγορίες αυτών. Στο δεύτερο κεφάλαιο, αναλύεται το django, η λειτουργικότητα αυτού και παραδείγματα εφαρμογών που έχουν υλοποιηθεί με αυτό. Στο τρίτο κεφάλαιο περιγράφεται η αναγκαιότητα μιας διαδικτυακής εφαρμογής σαν αυτή που υλοποιήθηκε , η λειτουργικότητά της, η αλληλεπίδραση με τον server και η βάση δεδομένων. Κατά την περιγραφή των παραπάνω δίνονται και αντιπροσωπευτικά κομμάτια κώδικα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ : Ανάπτυξη διαδικτυακής εφαρμογής

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : framework, django, μουσικοί, διαδίκτυο, υπερπληροφόρηση

Περιεχόμενα

ΠΕΡΙΛΗΨΗ.....	3
ΕΙΣΑΓΩΓΗ.....	5
ΚΕΦΑΛΑΙΟ 2.....	7
2.1 Τι είναι τα software frameworks	7
2.2 Τι είναι τα web frameworks και γιατί χρειάζονται.....	8
2.2.1 Χαρακτηριστικά ενός web framework.....	8
2.2.2 Αρχιτεκτονική των web frameworks.....	10
2.2.3 Server-side frameworks	12
ΚΕΦΑΛΑΙΟ 3.....	16
3.1 Διαδικασία εγκατάστασης του Django	16
3.1.1 Διαδικασία εγκατάστασης της Python	16
3.1.2 Στήσιμο της βάσης δεδομένων.....	17
3.1.3 Εγκατάσταση του Django.....	17
3.2 Επισκόπηση της λειτουργικότητας του django	20
3.2.1 Σχεδιασμός μοντέλων της βάσης δεδομένων	20
3.2.2 Αξιοποίηση του API.....	21
3.2.3 Admin Interface	25
3.2.4 Σχεδιασμός των URLs.....	26
3.2.4 Σχεδιασμός των Views	27
3.2.5 Σχεδιασμός των Templates.....	27
ΚΕΦΑΛΑΙΟ 4.....	29
4.1 Φιλοσοφία και αρχικός σχεδιασμός της εφαρμογής.....	29
4.1.2 Φιλοσοφία	29
4.1.2 Αρχικός σχεδιασμός της εφαρμογής	30
4.2 Τρόπος εγκατάστασης και περιγραφή των λειτουργιών του project σε αρχάριο επίπεδο	36
4.2.1 Επιλογή εργαλείου για την υλοποίηση της εφαρμογής.....	36
4.2.2 Εγκατάσταση του Django για την υλοποίηση της εφαρμογής.....	37
4.2.3 Δημιουργία του project και του app	37
4.3 Περιγραφή μοντέλων και λειτουργιών της τελειοποιημένης εφαρμογής.....	44
4.3.1 Εκκίνηση του server	50
4.3.2 Home_page πριν την εγγραφή	51
4.3.3 Sign_up page.....	53

4.3.4 Home page μετά την εγγραφή.....	56
4.3.5 Profile page	57
4.3.6 Song list page	62
4.3.7 Profile list page.....	66
4.3.8 Profile appearance page	68
4.3.9 Other song list page	69
4.3.10 Other lyrics list page	70
4.3.11 Messaging	72
4.3.12 Logout/Login	75
ΚΕΦΑΛΑΙΟ 5	76
5.1 Συμπεράσματα και μελλοντικές επεκτάσεις	76
ΚΕΦΑΛΑΙΟ 6	77
6.1 Δικτυογραφία.....	77

ΕΙΣΑΓΩΓΗ

Από τα τέλη του δεύτερου μισού του 20ου αιώνα έως σήμερα ζούμε στην εποχή της πληροφορίας. Η εποχή αυτή χαρακτηρίζεται από την δυνατότητα του ανθρώπου να μπορεί μέσω τεχνολογιών να διανείμει ή να έχει πρόσβαση σε πληροφορία που θα ήταν δύσκολο ή αδύνατο να βρει χωρίς τις τεχνολογίες αυτές. Και επειδή οι τεχνολογίες αυτές έχουν χαμηλό κόστος για τον μέσο καταναλωτή και η χρήση τους δεν απαιτεί ιδιαίτερα εξειδικευμένες γνώσεις, μια πολύ μεγάλη μερίδα ανθρώπων τις χρησιμοποιεί με σκοπό να εισάγει ή να ζητήσει καθημερινά πληροφορία από το διαδίκτυο με αποτέλεσμα την συνεχή και ραγδαία αύξηση της πληροφορίας.

Αυτή η πραγματικότητα έχει τόσο θετικές όσο και αρνητικές επιπτώσεις. Οι θετικές επιπτώσεις σχετίζονται με το ότι αυξάνεται η ποικιλία της πληροφορίας που διανέμεται με αποτέλεσμα να μπορεί ο χρήστης του διαδικτύου να έχει πρόσβαση σε όλο και μεγαλύτερο εύρος πληροφοριών. Οι αρνητικές επιπτώσεις σχετίζονται με δύο περιπτώσεις, την αύξηση της παραπληροφόρησης και τη δυσκολία εύρεσης συγκεκριμένων πληροφοριών. Στη πρώτη περίπτωση, επειδή όπως προαναφέρθηκε είναι πολύ μεγάλος ο αριθμός των χρηστών του διαδικτύου, αυτομάτως αυξάνεται και η πιθανότητα εισαγωγής ψευδών στοιχείων πάνω σε κάποιο θέμα από χρήστες, καθώς και η αναπαραγωγή αυτών είτε με συγκεκριμένο σκοπό είτε από αφέλεια από άλλους χρήστες. Στη δεύτερη περίπτωση, επειδή αυξάνεται με μεγάλη ταχύτητα ο όγκος της πληροφορίας αυξάνεται και η δυσκολία ενός χρήστη να βρει κάτι συγκεκριμένο.

Ένας τρόπος αντιμετώπισης της δυσκολίας της δεύτερης περίπτωσης είναι το φιλτράρισμα των πληροφοριών μέσω διαδικτυακών εφαρμογών. Με άλλα λόγια, η δημιουργία διαδικτυακών εφαρμογών όπου η κάθε μία θα έχει μια θεματική έτσι ώστε οι ενδιαφερόμενοι χρήστες να μπορούν πιο ευκολά να βρουν κάτι συγκεκριμένο πάνω σε αυτήν τη θεματική. Για παράδειγμα αν κάποιος θέλει να βρει πληροφορίες σχετικά με τη καλλιέργεια ενός συγκεκριμένου φυτού θα αναζητήσει μέσω της μηχανής αναζήτησης της επιλογής του κάποια διαδικτυακή εφαρμογή που να έχει ως θεματική τη καλλιέργεια των φυτών και μέσω αυτής θα ψάξει τις πληροφορίες που θέλει.

Υπάρχουν διάφοροι τρόποι κατασκευής διαδικτυακών εφαρμογών. Ένας πολύ πρακτικός τρόπος είναι μέσω κάποιων frameworks τα οποία θα αναλυθούν παρακάτω.

Στη συγκεκριμένη πτυχιακή εργασία αναπτύχθηκε ολοκληρωμένη διαδικτυακή εφαρμογή με θεματική την on line δημιουργία μουσικών κοινωνιών. Οι χρήστες της μπορούν να δημιουργούν προφίλ σύμφωνα με τα προσωπικά τους στοιχεία(Όνομα χρήστη, Όνομα, Επίθετο κλπ) τα μουσικά ενδιαφέροντά τους (Rock, Blues κλπ) και τις μουσικές ιδιότητές τους (Τραγουδιστής, Ντράμερ κλπ). Συγκεκριμένα, για την αναζήτηση θα πρέπει να συμπληρωθούν ή το Όνομα χρήστη ή τα Μουσικά ενδιαφέροντα ή οι Μουσικές ιδιότητες ώστε μπορούν να βρίσκουν ο ένας τον άλλο. Για παράδειγμα μπορεί μια μπάντα που θέλει να παίξει Blues-Rock και Metal να μην είναι ολοκληρωμένη διότι της λείπει ο ντράμερ ο οποίος πρέπει και να έχει τα ίδια μουσικά ενδιαφέροντα. Μέσω της παραπάνω εφαρμογής μπορούν τα μέλη της μπάντας να αναζητήσουν ένα μουσικό προφίλ με βάση τα παρακάτω στοιχεία:

Μουσικά Ενδιαφέροντα: Blues-Rock, Metal

Μουσικές Ιδιότητες : Ντράμερ

Αν υπάρχει συνδυασμός αυτών των στοιχείων σε κάποιο προφίλ τότε αυτό εμφανίζεται και μπορούν τα μέλη της μπάντας να επικοινωνήσουν μαζί του.

ΚΕΦΑΛΑΙΟ 2

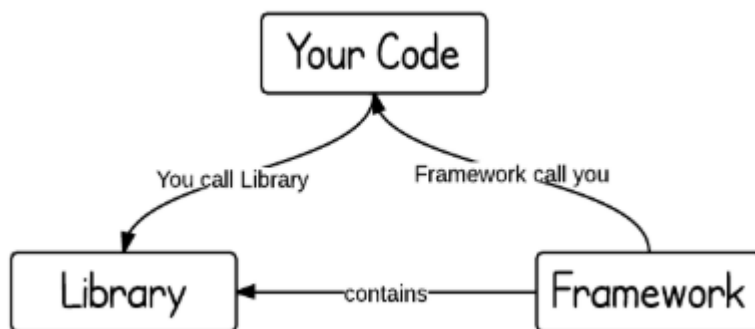
2.1 Τι είναι τα software frameworks

Πριν αναλύσουμε τι είναι τα web frameworks (WFs) ή web application frameworks (WAFs) θα εξηγήσουμε πρώτα, για την καλύτερη κατανόησή τους, την κατηγορία στην οποία ανήκουν δηλαδή τα software frameworks. Ένα software framework αποτελεί πλατφόρμα με επαναχρησιμοποιούμενο κώδικα με σκοπό την διευκόλυνση της ανάπτυξης εφαρμογών λογισμικού. Για τον σκοπό αυτό παρέχει στον χρήστη του βιβλιοθήκες κώδικα, βοηθητικά προγράμματα, compilers, προγραμματιστικά εργαλεία, και API's που διευκολύνουν την ροή των δεδομένων. Βασικά χαρακτηριστικά που διαφοροποιούν ένα software framework από τις κανονικές βιβλιοθήκες είναι τα εξής :

Inversion of control: Η πιο βασική διαφορά μεταξύ ενός software framework και μιας βιβλιοθήκης σχετίζεται με την αντιστροφή στον καθορισμό του ελέγχου ροής προγράμματος (flow of control ή control flow).

Μια βιβλιοθήκη είναι απλά συλλογή από έτοιμα υποπρογράμματα. Ο λόγος που υπάρχουν είναι για την απλή επαναχρησιμοποίηση κώδικα. Με άλλα λόγια, να μπορεί κάποιος προγραμματιστής να χρησιμοποιήσει έτοιμο κώδικα που έχει γραφτεί από άλλους προγραμματιστές, καλώντας για παράδειγμα μια μέθοδο από μια βιβλιοθήκη. Η μέθοδος αυτή ουσιαστικά καλείται από ένα πρόγραμμα του οποίου ο έλεγχος ροής έχει καθοριστεί από τον κώδικα.

Σε ένα software framework γίνεται το αντίθετο. Ο έλεγχος ροής υπάρχει ήδη στο software framework και υπάρχουν κάποια προκαθορισμένα σημεία στα οποία θα πρέπει να συμπληρωθούν με κώδικά ο οποίος θα καλεστεί όταν χρειαστεί.



Εικόνα 1-Inversion of control. Πηγή : <https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/>

Άρα, στην περίπτωση της βιβλιοθήκης, τον έλεγχο ροής τον έχει ο προγραμματιστής ενώ στην περίπτωση του software framework τον έλεγχο ροής έχει το ίδιο το framework

Extensibility: Ένας χρήστης μπορεί να επεκτείνει το software framework συμπληρώνοντας κώδικα στον default κωδικα με σκοπό την παροχή πιο εξειδικευμένης λειτουργικότητας από αυτή που παρέχεται ήδη.

Non-modifiable framework code: Ο κώδικας του software framework μπορεί να επεκταθεί αλλά δεν μπορεί να τροποποιηθεί.

2.2 Τι είναι τα web frameworks και γιατί χρειάζονται

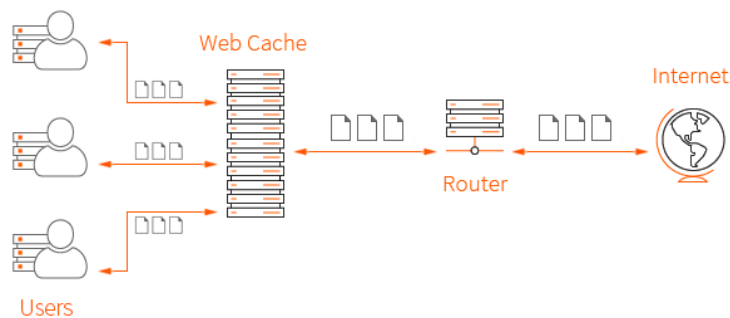
Τα web frameworks βασίζονται στην παραπάνω λογική προσαρμόζοντάς την στο να βοηθήσουν τον σχεδιασμό και την ανάπτυξη κυρίως δυναμικών διαδικτυακών εφαρμογών ή αλλιώς ιστοσελιδών με πληροφορία που αλλάζει αυτόματα. Παίζουν πολύ σημαντικό ρόλο όσον αφορά το web development διότι στην σημερινή μας εποχή υπάρχει η ανάγκη , όσο ποτέ άλλοτε , για την ανάπτυξη δυναμικών εφαρμογών. Ο λόγος για την ανάγκη αυτή είναι ότι οι απαιτήσεις των χρηστών σχετίζονται με την αλληλεπίδραση με τις εφαρμογές αυτές.

2.2.1 Χαρακτηριστικά ενός web framework

Ένα web framework αποτελείται από τα εξής χαρακτηριστικά

1. Web Cache

Ένα web cache είναι μια hardware συσκευή ή μια software εφαρμογή και χρησιμοποιείται για την προσωρινή αποθήκευση static περιεχομένου στο οποίο οι χρήστες έχουν συχνή πρόσβαση. Ουσιαστικά διαχειρίζεται requests για δημοφιλές περιεχόμενο το οποίο θα τα διαχειριζόταν ο server. Έτσι αποφεύγεται το φαινόμενο του server overloading.



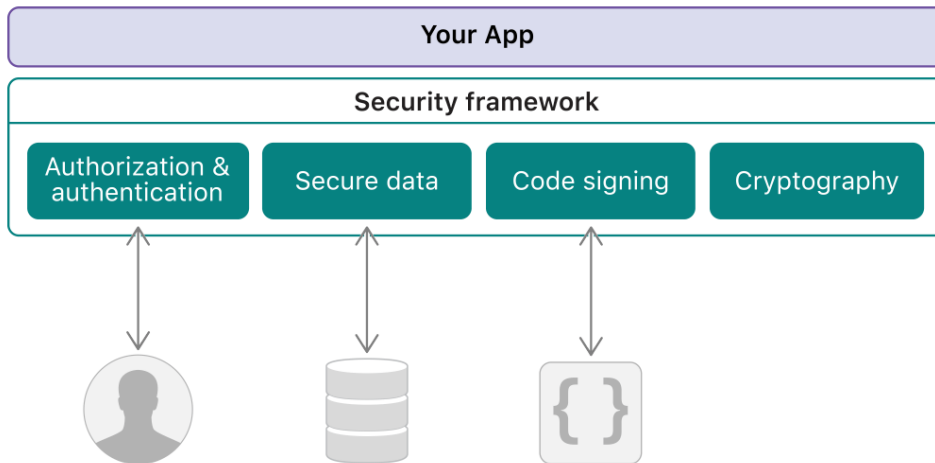
Εικόνα 2: Web Cache. Πηγή: <https://www.maxcdn.com/one/visual-glossary/web-cache/>

2. Web Templates

Τα web templates είναι προσχεδιασμένα templates τα οποία μπορούν εύκολα να επεξεργαστούν. Ουσιαστικά, ένα web template δίνει την δυνατότητα στον designer να εισάγει περιεχόμενο σε μια προσχεδιασμένη δομή φτιαγμένη στις γλώσσες HTML και CSS. Ένα web template μπορεί να περιέχει τίτλους σε συγκεκριμένες θέσεις, εικόνες, background και άλλα σχεδιαστικά χαρακτηριστικά του template, καθώς και σχεδιασμένη τυπογραφία. Οι designers μπορούν, χωρίς τη χρήση κώδικα, να εκμεταλλευτούν την παραπάνω δομή και να εισάγουν τα δικά τους δεδομένα (εικόνες, κείμενο κλπ) τα οποία θα προσαρμόζονται στο πώς είναι φτιαγμένη. Χαρακτηριστικό παράδειγμα web template είναι τα themes.

3. Security

Το online security έχει πολλά κριτήρια για να την ταυτοποίηση και την αποδοχή ή την απόρριψη πρόσβασης σε διαφορετικές λειτουργίες στο web framework. Επίσης βοηθάει στην αναγνώριση των χρηστών που χρησιμοποιούν την εφαρμογή ώστε να αποφευχθεί το φαινόμενο clickjacking (τεχνική εξαπάτησης ενός χρήστη ο οποίος κάνει κλικ σε κάτι διαφορετικό από αυτό που αντιλαμβάνεται το οποίο μπορεί να αποκαλύψει σε άλλους πληροφορίες "εμπιστευτικότητας" σχετικά με αυτόν.). Σαν αποτέλεσμα, το web framework, σχεδιασμένο από ειδικούς, εγγυάται σε μεγαλύτερο βαθμό τις διαδικασίες ταυτοποίησης και εξουσιοδότησης.



Εικόνα 3: Framework Security. Πηγή: <https://developer.apple.com/documentation/security>

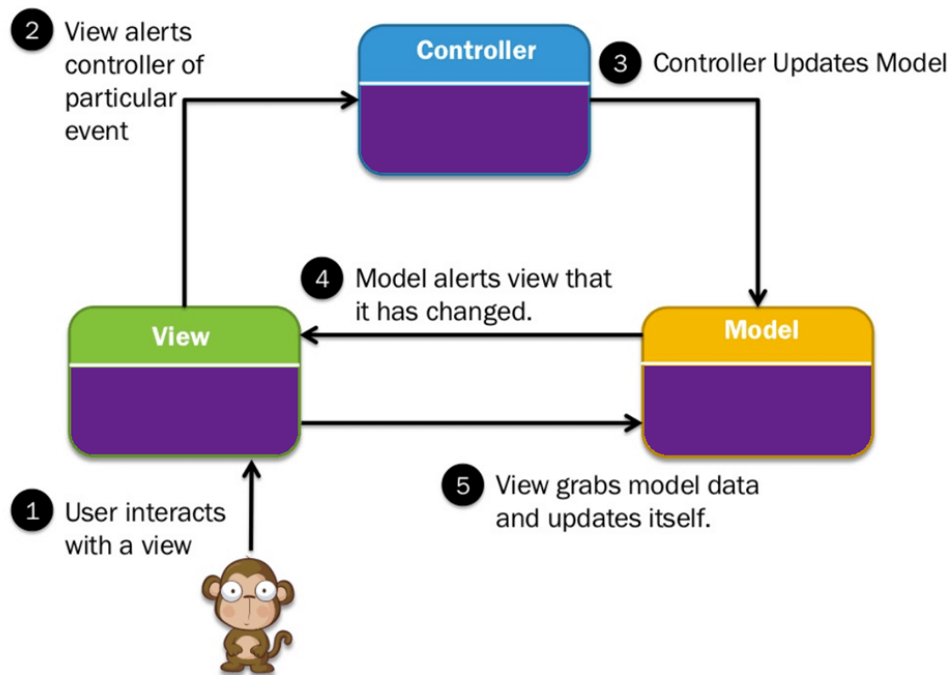
4. UrlMapping

Το url mapping προσφέρει τη δυνατότητα αντιστοίχισης των URL με συναρτήσεις που θα καλεστούν από αυτά τα URL. Ουσιαστικά παρέχει τη δυνατότητα κατασκευής προτύπων που αποτελούνται από path expressions με παραμέτρους που λαμβάνουν τιμές από το URL που ζητά ο χρήστης και καλούν την αντίστοιχη συνάρτηση περνώντας της τις τιμές αυτές. Οι συναρτήσεις αυτές έχουν σκοπό την αναπαράσταση μιας σελίδας.

Πολλών ειδών διαδικτυακών εφαρμογών στηρίζονται σε web frameworks. Τα πιο χρησιμοποιημένα web frameworks υποστηρίζουν την κατασκευή blog, forum, ιστοσελίδες με γενικό περιεχόμενο κλπ.

2.2.2 Αρχιτεκτονική των web frameworks

Οι αρχιτεκτονικές των περισσότερων web framework βασίζονται στο πρότυπο model-view-controller (MVC). Τα MVC χρησιμοποιούνται συνήθως για την ανάπτυξη user interfaces (UI) που χωρίζουν μια εφαρμογή σε τρία διασυνδεδεμένα μέρη. Αυτό γίνεται έτσι ώστε να διαχωριστεί η εσωτερική αναπαράσταση της πληροφορίας στην εφαρμογή από τον τρόπο με τον οποίο η πληροφορία παρουσιάζεται στον χρήστη έτσι ώστε να είναι user-friendly. Επίσης, διαχωρίζουν τα τρία βασικά συστατικά τους κάνοντας εφικτή την επαναχρησιμοποίηση του λογισμικού. Παρακάτω γίνεται σύντομη περιγραφή των συστατικών αυτών.



Εικόνα 4: MCV αρχιτεκτονική. Πηγή: <https://www.guru99.com/mvc-tutorial.html>

1. Model

Το model είναι το κεντρικό συστατικό του παραπάνω προτύπου. Αποτελεί την δυναμική δομή δεδομένων της εφαρμογής ανεξάρτητα από το user interface και διαχειρίζεται τα δεδομένα, τα business logic layers και τα rules της εφαρμογής

2. View

Το view είναι υπεύθυνο για τις οπτικές αναπαραστάσεις των δεδομένων όπως διαγράμματα, πίνακες κλπ

3. Controller

Μετατρέπει τα input data σε εντολές που μπορούν να εφαρμοστούν στο model ή στο view.

Ο τρόπος με τον οποίο έχει καθοριστεί από το MVC να αλληλοεπιδρούν τα συστατικά του είναι ο εξής:

Το model είναι υπεύθυνο για την διαχείριση των δεδομένων της εφαρμογής. Το user input στο model φτάνει μέσω του controller

Το view σημαίνει αναπαράσταση του μοντέλου με μια συγκεκριμένη μορφή

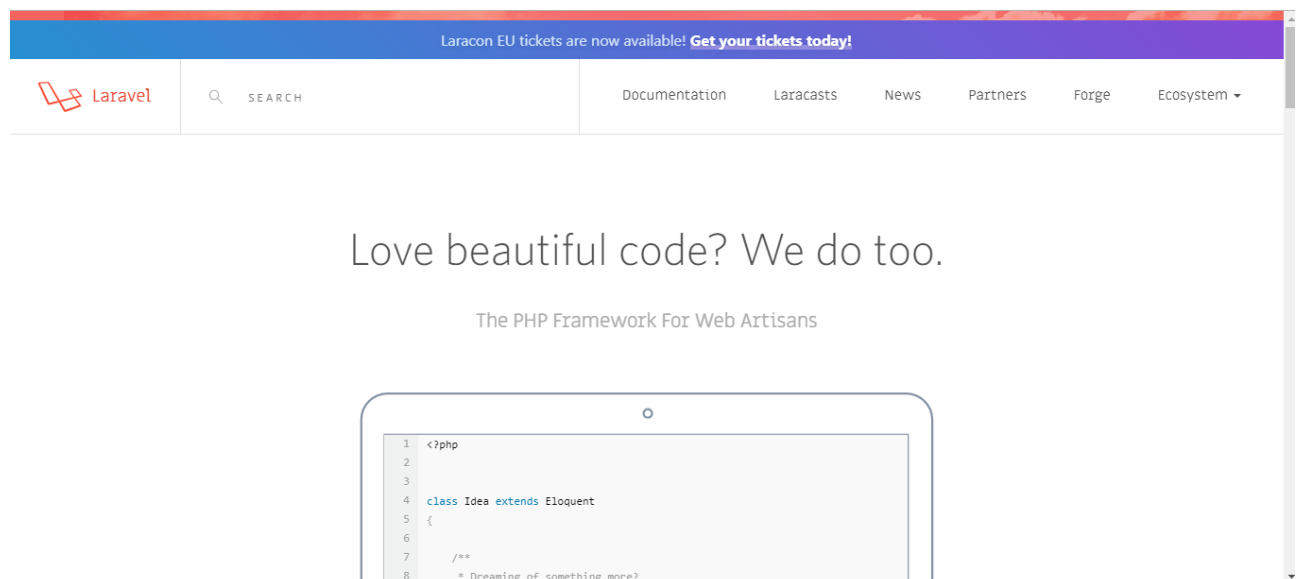
Ο controller λαμβάνει το input, το επικυρώνει και μετά το περνά στο model

2.2.3 Server-side frameworks

Τα server-side frameworks σχετίζονται με το business logic (δηλαδή το μέρος του προγράμματος που καθορίζει το πώς τα δεδομένα αποθηκεύονται, δημιουργούνται και αλλάζουν) Μέσω αυτών των framework μπορούν να δημιουργηθούν απλές σελίδες, landing σελίδες και φόρμες διαφορετικών τύπων. Όμως, δεν παρέχουν τη δυνατότητα για την δημιουργία ενός καλά ανεπτυγμένου User Interface της διαδικτυακής εφαρμογής καθώς για αυτό απαιτείται παραπάνω λειτουργικότητα. Επίσης, μπορούν να βελτιώσουν την ασφάλεια σε περίπτωση διαδικτυακών επιθέσεων.

Στα πιο χρησιμοποιημένα server-side frameworks(ή backend frameworks) ανήκουν τα παρακάτω:

1. Laravel



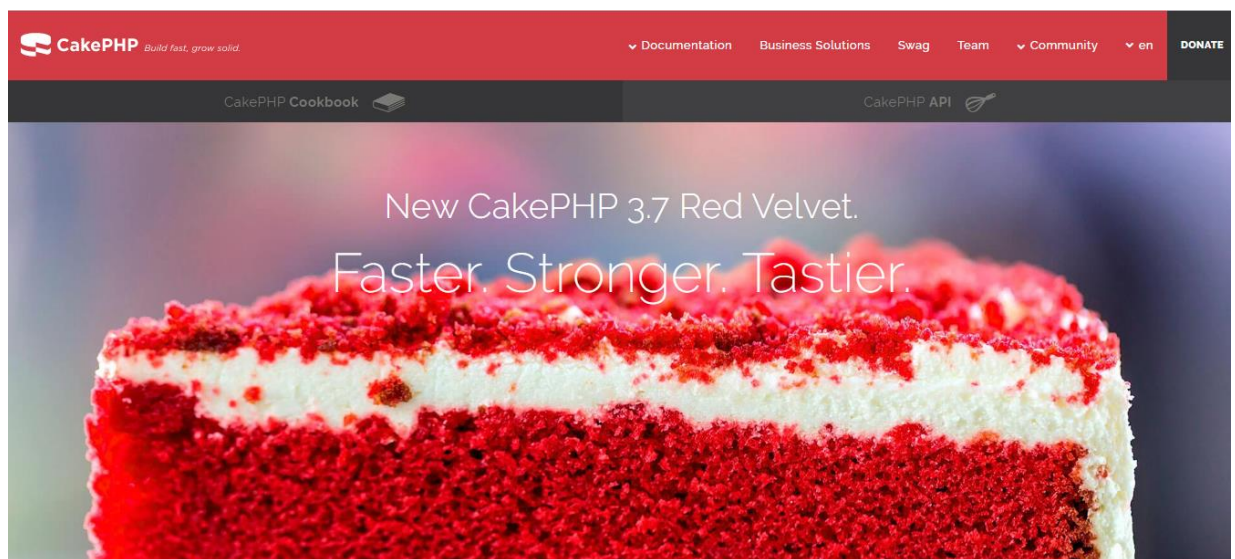
Εικόνα 5: Laravel. Πηγή: <https://laravel.com/>

Το Laravel βασίζεται στην γλώσσα PHP. Ξεχωρίζει για τη δυνατότητα να χρησιμοποιείται από μεγάλες ομάδες για πολύπλοκες εφαρμογές όπως εφαρμογές για επιχειρήσεις, και την πλούσια λειτουργικότητα της εργαλειοθήκης (toolkit) που παρέχει. Στηρίζεται στο πρότυπο MVC και παρέχει το δικό του database migration system.

Χαρακτηριστικά:

- Απλή και γρήγορη routing engine
- Έχει δικό του command line interface
- Ισχυρό template system
- Πλούσιο documentation

2. CakePHP



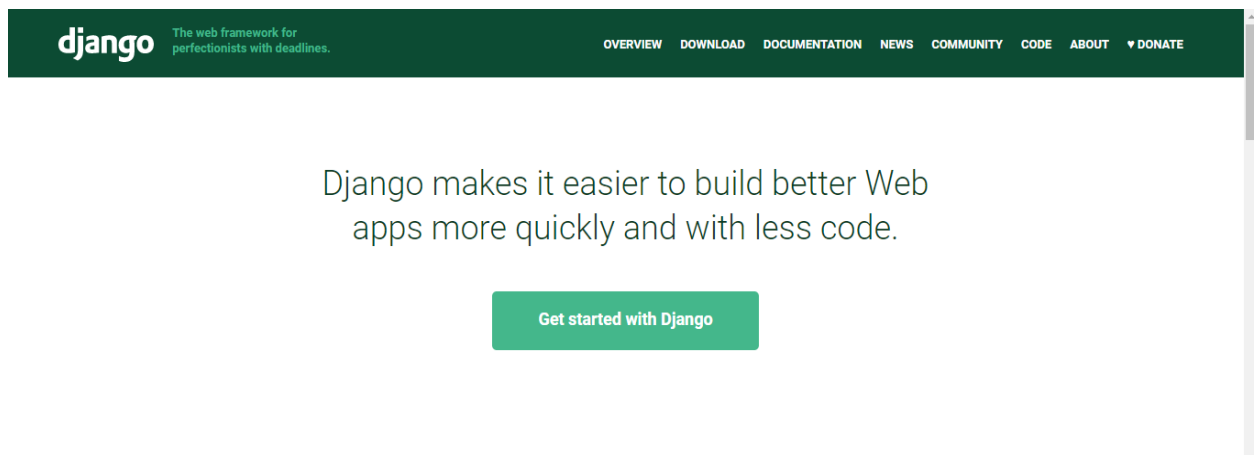
Εικόνα 6: CakePHP. Πηγή: <https://cakephp.org/>

Το CakePHP είναι ένα από τα πρώτα frameworks βασισμένα σε PHP. Δημιουργήθηκε το 2005 και από τότε έχει εξελιχθεί αρκετά ώστε να θεωρείται σήμερα ένα σύγχρονο web framework που επιτρέπει στους χρήστες του να δημιουργήσουν website σχετικά γρήγορα. Επίσης, στηρίζεται στο MVC πρότυπο και παρέχει την δυνατότητα επέκτασής του δίνοντας την δυνατότητα ανάπτυξης και απλών αλλά και πολύπλοκων εφαρμογών.

Χαρακτηριστικά

- Έχει πληθώρα ενσωματωμένων συναρτήσεων και κλάσεων που καλύπτουν τις πιο κοινές απαιτήσεις που χρειάζονται για την ανάπτυξη διαδικτυακών εφαρμογών.
- Γρήγορη ανάπτυξη διαδικτυακών εφαρμογών
- Ασφάλεια
- Δεν χρειάζονται πολύπλοκες ρυθμίσεις για την εκκίνησή του

3. Django



Εικόνα 7: Django. Πηγή: <https://www.djangoproject.com/>

Το Django είναι framework της python το οποίο δημιουργήθηκε με την λογική “batteries included” δηλαδή να παρέχει στον web developer ό,τι χρειάζεται περιορίζοντας την ανάγκη για εξωτερικές βιβλιοθήκες. Για τον λόγο αυτό, το Django είναι κατάλληλο για την ανάπτυξη πολύπλοκων εφαρμογών. Στην περίπτωση ανάπτυξης μια απλής εφαρμογής το django δεν είναι κατάλληλο διότι θα επιβαρύνει την εφαρμογή με επιπλέον λειτουργίες της οποίες δεν θα αξιοποιεί.

Χαρακτηριστικά

- Προσαρμόζεται εύκολα στις απαιτήσεις του web developer
- Γρήγορος προγραμματισμός
- Μπορεί να επεκταθεί σε μεγάλο βαθμό
- Πλούσιο documentation μεγάλο community

Στο επόμενο κεφάλαιο γίνεται εκτενέστερη περιγραφή των λειτουργιών του Django

4. Flask



[overview](#) // [docs](#) // [community](#) // [extensions](#) // [donate](#)

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's [BSD licensed](#)!

Εικόνα 8: Flask. Πηγή: <http://flask.pocoo.org/>

Το Flask βασίζεται στην γλώσσα python. Σε αντίθεση με το Django, παρέχει λιγότερες λειτουργίες κάτι το οποίο το καθιστά κατάλληλο για την ανάπτυξη απλών εφαρμογών. Υποστηρίζει το Jinja templating(μια designer-friendly γλώσσα μέσω της οποία μπορούν να εισαχθούν εντολές της python στα templates του framework) , secure cookies(δηλαδή cookies με περιορισμένο πεδίο εφαρμογής σε “ασφαλή ” κανάλια (HTTPS)), unit testing (κομμάτια κώδικα τα οποία εξετάζουν αν μια κλάση ή μια συνάρτηση έχει ομαλή συμπεριφορά) καθώς και δικό του Restful API, δηλαδή τα βασικά HTTP requests (GET για την ανάκτηση δεδομένων, PUT για την ανανέωση δεδομένων, POST για την δημιουργία δεδομένων και DELETE για την διαγραφή δεδομένων).

Χαρακτηριστικά

- Πολύ ευέλικτο
- Ιδανικό για απλές εφαρμογές
- Πλούσιο documentation
- Προσφέρει τη δυνατότητα για την γρήγορη δημιουργία προτύπων ιστοσελίδας

ΚΕΦΑΛΑΙΟ 3

3.1 Διαδικασία εγκατάστασης του Django

3.1.1 Διαδικασία εγκατάστασης της python

Επειδή το Django είναι web framework της python είναι απαραίτητη η εγκατάστασή της για την λειτουργία του.

3.1.1.1 Έκδοση της Python

Η έκδοση που προτείνεται από το documentation του Django είναι η τελευταία έκδοση της python 3 διότι ,σε σχέση με παλαιότερες εκδόσεις, είναι γρηγορότερη, έχει περισσότερα χαρακτηριστικά και υποστηρίζεται πιο εύκολα. Το Django λειτουργεί και με παλαιότερες εκδόσεις.

3.1.1.2 Εγκατάσταση της python

Μέσω του link <https://www.python.org/downloads/> οδηγούμαστε στην ιστοσελίδα από την οποία μπορούμε να κατεβάσουμε την έκδοση της python που θέλουμε για το αντίστοιχο λειτουργικό σύστημα που χρησιμοποιούμε και ύστερα να κάνουμε την εγκατάσταση. Ένας εναλλακτικός τρόπος για την εγκατάσταση της python είναι μέσω του package manager(συλλογή από software tools τα οποία αυτοματοποιούν την διαδικασία εγκατάστασης, αναβάθμισης, παραμετροποίησης και διαγραφής προγραμμάτων για ένα λειτουργικό σύστημα) του λειτουργικού μας συστήματος. Για να επιβεβαιώσουμε την εγκατάσταση της python , γράφουμε python στην command line και μας εμφανίζεται κάτι σαν το παρακάτω:

```
C:\Users\user>python
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

Εικόνα 9: Python version

3.1.2 Στήσιμο της βάσης δεδομένων

Η python έχει ενσωματωμένη την SQLite που είναι ιδανική για απλές εφαρμογές και δεν χρειάζεται κάποιον ξεχωριστό server για να λειτουργήσει. Στην περίπτωση όμως που η εφαρμογή μας απαιτεί μια “μεγάλη” βάση δεδομένων θα χρειαστεί να δουλέψουμε μέσω του αντίστοιχου database server (υπολογιστικό σύστημα που παρέχει σε άλλους υπολογιστές υπηρεσίες σχετικά με την πρόσβαση και ανάκτηση δεδομένων από μια βάση). Οι database servers που υποστηρίζει επίσης το Django εκτός από την ενσωματωμένη SQLite- είναι η PostgreSQL, η MySQL και η Oracle όπου ο κάθε ένας χρειάζεται την εγκατάσταση του αντίστοιχου Package για να λειτουργήσει. Το psycopg2 για την PostgreSQL, έναν DB API driver όπως ο mysqlclient για την MySQL, και τον driver cx_Oracle για την Oracle. Η εγκατάσταση των packages γίνεται μέσω του python Package Installer (pip). Το pip είναι ήδη εγκατεστημένο στις εκδόσεις python 2 >=2.7.9 ή python 3 >=3.4. Στην περίπτωση που δεν είναι εγκατεστημένο μπορούμε να το κατεβάσουμε μέσω του command line με την εντολή

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py  
και να το εγκαταστήσουμε με την εντολή python get-pip.py
```

```
python get-pip.py
```

.

Εφόσον επιβεβαιώσουμε ότι το pip είναι εγκατεστημένο, μπορούμε να εγκαταστήσουμε το package που θέλουμε γράφοντας για παράδειγμα στο command line την εντολή

```
pip install psycopg2
```

Εκτός από τους επίσημους database servers που υποστηρίζει το Django, μπορούν να χρησιμοποιηθούν μέσω third-parties οι παρακάτω :

IBM DB2, Microsoft SQL Server, Firebird, ODBC

3.1.3 Εγκατάσταση του Django

Ο πιο διαδεδομένος τρόπος εγκατάστασης του django είναι μέσω του pip

3.1.3.1 Εγκατάσταση του pip

Εγκαθιστούμε το pip(βλ 3.1.2). Εάν είναι ήδη εγκατεστημένο επιβεβαιώνουμε εάν η έκδοσή του δεν είναι ξεπερασμένη. Για να το κάνουμε αυτό μπορούμε στο command line να τρέξουμε την εντολή

```
pip --version
```

η οποία μας εμφανίζει την έκδοση της pip. Εάν είναι μικρότερη από την πιο πρόσφατη (την οποία μπορούμε να δούμε στο link <https://pypi.org/project/pip/>) τότε μπορούμε να την αναβαθμίσουμε με την εντολή

```
pip install -U pip
```

για Linux και Mac και την εντολή

```
python -m pip install -U pip
```

για Windows

3.1.3.2 Εγκατάσταση ενός virtual enviroment

Ένα virtual enviroment είναι οποιαδήποτε εφαρμογή επιτρέπει τόσο την αλληλεπίδραση με το computing enviroment (συλλογή από υπολογιστές/μηχανήματα, λογισμικά και δίκτυα που υποστηρίζουν την ανταλλαγή και την επεξεργασία ηλεκτρονικής πληροφορίας με σκοπό την επίλυση υπολογιστικών προβλημάτων) όσο και το έργο άλλων χρηστών (πχ email, chat) Στην python, αποτελεί εργαλείο που βοηθά στο να μένουν ξεχωριστά τα dependencies (βιβλιοθήκες ξέχωρες από τον κώδικα ενός project που είναι απαραίτητες για την δημιουργία και την εκκίνηση μιας εφαρμογής) του κάθε project δημιουργώντας απομονωμένα python virtual enviroments για το κάθε project. Ο λόγος που χρειάζονται είναι για να μπορούμε να δουλέψουμε με πολλαπλά projects από διαφορετικές εκδόσεις του django. Αν για παράδειγμα αναπτύσσουμε παράλληλα ένα project με την έκδοση django 9.0 και ένα άλλο με την έκδοση django 10.0 χρειαζόμαστε ένα virtual enviroment για το κάθε ένα ειδάλλως θα αποθηκεύουν στον ίδιο φάκελο και θα υπάρξει πρόβλημα στη λειτουργικότητά τους. Και επειδή ένα virtual enviroment ουσιαστικά αποτελεί έναν φάκελο με λίγα scripts , δεν υπάρχει όριο στο πόσα μπορούμε να φτιάξουμε.

Για να δημιουργήσουμε λοιπόν virtual enviroments χρησιμοποιούμε ένα module, το virtualenv. Για να το εγκαταστήσουμε, τρέχουμε την εντολή

```
pip install virtualenv
```

Για να δούμε την έκδοσή του τρέχουμε την εντολή

```
virtualenv --version
```

Για να δημιουργήσουμε ένα virtualenv με όνομα name τρέχουμε την εντολή

```
virtualenv name
```

.Αυτή η εντολή θα δημιουργήσει έναν φάκελο με όνομα name που περιέχει όλα τα εκτελέσιμα προγράμματα που χρειάζονται για την χρήση των packages που είναι απαραίτητα για ένα python project. Αν θέλουμε να καθορίσουμε τον διερμηνέα της αντίστοιχης έκδοσης της Python που θα χρησιμοποιήσουμε (πχ python3) τρέχουμε την εντολή

```
virtualenv -p /usr/bin/python3 name
```

Αφού δημιουργήσουμε το virtual enviroment, θα πρέπει να το ενεργοποιήσουμε. Αυτό γίνεται με την εντολή

```
source virtualenv_name/bin/activate
```

Να σημειωθεί ότι το virtual enviroment πρέπει να ενεργοποιείται κάθε φορά που θέλουμε να δουλέψουμε με το αντίστοιχο project.

3.1.3.3 Εγκατάσταση της επιθυμητής έκδοσης του Django

Αφού έχουμε εγκαταστήσει και ενεργοποιήσει το virtual enviroment(το οποίο θα φαίνεται στα αριστερά του command line) , μπορούμε να εγκαταστήσουμε την έκδοση του django(πχ 1.9) που θέλουμε με την εντολή

```
(name)$ pip install Django==1.9
```

Αν θέλουμε την τελευταία έκδοση του django :

```
$ pip install Django
```

Για να απενεργοποιήσουμε το virtual enviroment ώστε να επαναφέρουμε τις ρυθμίσεις της Python τρέχουμε την εντολή:

```
(virtualenv_name)$ deactivate
```

3.1.3.4 Επιβεβαίωση επιτυχούς εγκατάστασης

Για να επιβεβαιώσουμε την επιτυχή εγκατάσταση του django γράφουμε στην command line

```
python -m django --version
```

Αν η εγκατάσταση ήταν επιτυχής θα πρέπει να εκτυπώνεται η έκδοση του django που εγκαταστήσαμε αλλιώς θα μας εμφανίσει “No module named django”

3.2 Επισκόπηση της λειτουργικότητας του django

Στόχος της παρακάτω επισκόπησης είναι να επισημανθούν οι τεχνικές ιδιαιτερότητες του django με σκοπό την κατανόηση του πώς δουλεύει.

3.2.1 Σχεδιασμός μοντέλων της βάσης δεδομένων

Μπορούμε να διαμορφώσουμε την βάση δεδομένων με python κώδικα. Επιλέγουμε το αρχείο στο οποίο θα περιέχονται τα μοντέλα μας (models.py) κάνουμε Import τα models και σχεδιάζουμε την βάση. Για παράδειγμα:

```
from django.db import models

class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __str__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)

    def __str__(self):
        return self.headline
```

Εικόνα 10: Django models. Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Αφού σχεδιάσουμε τα μοντέλα μας μπορούμε να φτιάξουμε τους πίνακες που αντιστοιχούν για το κάθε ένα στην βάση δεδομένων πρώτα με την εντολή

```
python manage.py makemigrations "όνομα της εφαρμογής"
```

και μετά με την εντολή

```
python manage.py migrate
```

3.2.2 Αξιοποίηση του API

Μπορούμε με απλές εντολές χρησιμοποιήσουμε το python API που παρέχει το Django ώστε να επεξεργαζόμαστε τα δεδομένα της βάσης μας. Στο παρακάτω παράδειγμα έχουμε φτιάξει μια εφαρμογή που περιέχει τα μοντέλα Reporter και Article (βρ 2.2.1) και τα επεξεργαζόμαστε μέσω του API:

Κάνουμε import τα μοντέλα μας:

```
>>> from news.models import Article, Reporter
```

Εικόνα 11: Model Import. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Με την συνάρτηση all() μπορούμε να δούμε όλα τα objects ενός model. Στην περίπτωση μας δεν υπάρχουν ακόμα “reporters” στο σύστημα:

```
>>> Reporter.objects.all()  
<QuerySet []>
```

Εικόνα 12: all() API. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Δημιουργούμε ένα αντικείμενο “reporter” βάζοντας στο field full_name την τιμή ‘John Smith’ :

```
>>> r = Reporter(full_name='John Smith')
```

Εικόνα 13: Reporter add. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Για να το αποθηκεύσουμε στη βάση χρησιμοποιούμε την συνάρτηση save():

```
>>> r.save()
```

Εικόνα 14: save() API. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Επειδή είναι το πρώτο αντικείμενο που φτιάξαμε έχει id 1:

```
>>> r.id  
1
```

Εικόνα 15: id check. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Τώρα υπάρχει ένας “reporter” στην βάση μας. (Ο λόγος που εμφανίζεται το full name ως αντιπροσωπευτικό για το object είναι επειδή το ορίσαμε παραπάνω στην συνάρτηση def.αλλιώς θα εμφάνιζε object(1)):

```
>>> Reporter.objects.all()  
<QuerySet [<Reporter: John Smith>]>
```

Εικόνα 16: Reporter object representation . **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Το πεδίο (full_name) παρουσιάζεται σαν παράμετρος του Python object που φτιάξαμε:

```
>>> r.full_name  
'John Smith'
```

Εικόνα 17: Reporter object full_name field representation. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Άλλες δυνατότητες που παρέχει το API του Django:

Εμφάνισε τον reporter με id=1:

```
>>> Reporter.objects.get(id=1)  
<Reporter: John Smith>
```

Εικόνα 18: Get API(id=1) . **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Εμφάνισε τους reporter που το όνομά τους αρχίζει με 'John';

```
>>> Reporter.objects.get(full_name__startswith='John')
<Reporter: John Smith>
```

Εικόνα 19: Get API(full_name__startswith) . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Εμφάνισε τους reporter που το όνομά τους περιέχει τη γραμματοσειρά 'mith'

```
>>> Reporter.objects.get(full_name__contains='mith')
<Reporter: John Smith>
```

Εικόνα 20: Get API(full_name__contains) . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Εμφάνισε τον reporter με id=2:

```
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
...
DoesNotExist: Reporter matching query does not exist.
```

Εικόνα 21: Get API(id=2) . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Τώρα θα δημιουργήσουμε ένα Article object :

```
>>> from datetime import date
>>> a = Article(pub_date=date.today(), headline='Django is cool',
...             content='Yeah.', reporter=r)
>>> a.save()
```

Εικόνα 22: Article create . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Στην `pub_date` εκχωρούμε την συνάρτηση `today()`. Η `today()` επιστρέφει τρέχουσα ημερομηνία ανάλογα με το `timezone` που έχουμε θέσει στις ρυθμίσεις. Στην `headline` κι στο `context` εκχωρούμε ένα κείμενο. Στον `reporter` (που είναι το foreign key στον πίνακα `Reporter`) εκχωρούμε `r` στο οποίο πριν εκχωρήσαμε τον “reporter” που δημιουργήσαμε

Πλέον το “article” βρίσκεται στην βάση:

```
>>> Article.objects.all()
<QuerySet [<Article: Django is cool>]>
```

Εικόνα 23: `Article.all()` . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Τα `Article` objects έχουν API πρόσβαση στα σχετιζόμενα `Reporter` objects:

```
>>> r = a.reporter
>>> r.full_name
'John Smith'
```

Εικόνα 24: `API(foreign_key)` . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Στην `r` εκχωρούμε τον “reporter” που σχετίζεται με το “article” α(το οποίο φτιάξαμε παραπάνω) και εμφανίζουμε το `full_name` του.

Μπορούμε και αντιστρόφως να εμφανίσουμε όλα τα “article” που σχετίζονται με τον παραπάνω “reporter”:

```
>>> r.article_set.all()
<QuerySet [<Article: Django is cool>]>
```

Εικόνα 25: `API(set)` . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Το API μπορεί κάνοντας τα κατάλληλα JOIN’s να μας εμφανίζει όλα τα “article” που σχετίζονται με τον “reporter” που το όνομά του αρχίζει από “John”:

```
>>> Article.objects.filter(reporter__full_name__startswith='John')
<QuerySet [<Article: Django is cool>]>
```

Εικόνα 26: API(filter) . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Μπορούμε να αλλάξουμε ένα object αλλάζοντας τις παραμέτρους του και αποθηκεύοντάς τες καλώντας την save() :

```
>>> r.full_name = 'Billy Goat'
>>> r.save()
```

Εικόνα 27: Modify field . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Για να διαγράψουμε ένα object καλούμε την συνάρτηση delete():

```
>>> r.delete()
```

Εικόνα 28: API(delete) . Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

3.2.3 Admin Interface

Αφού ορίσουμε τα μοντέλα μας στην βάση, μπορούμε να τα κάνουμε register στο administrative interface που παρέχει το Django, όπου μέσα σε αυτό θα μπορούν πιστοποιημένοι χρήστες να προσθέτουν, να αφαιρούν και να αλλάζουν objects με εύκολο τρόπο. Για να κάνουμε Register ένα μοντέλο (πχ το Article) επιλέγουμε το αρχείο στο οποίο θα περιέχονται registered μοντέλα μας (admin.py) και κάνουμε το εξής:

```
from django.contrib import admin

from . import models

admin.site.register(models.Article)
```

Εικόνα 29: Admin model register. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

3.2.4 Σχεδιασμός των URLs

Το Django βοηθάει στον σχεδιασμό καλαίσθητων URLs χωρίς να τους προσθέτει περιττές καταλήξεις όπως .php και .asp.

Για να σχεδιάσουμε URLs για την εφαρμογή μας δημιουργούμε ένα module της python , το URLconf. Παρακάτω φαίνεται ένα URLconf που σχετίζεται με τα μοντέλα που προαναφέραμε (Reporter/Article):

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<int:pk>/', views.article_detail),
]
```

Εικόνα 30: URL patterns. **Πηγή:** <https://docs.djangoproject.com/en/2.2/intro/overview/>

Ο παραπάνω κώδικας σχετίζει URL paths με callback συναρτήσεις της python που λέγονται views. Οι γραμματοσειρές των paths χρησιμοποιούν παραμέτρους (πχ 'articles/<int:year>/') για να συλλέξουν τιμές από το καταχωρημένο URL. Όταν ο χρήστης ζητήσει μια σελίδα μέσω ενός URL το Django τρέχει το κάθε ένα path ξεχωριστά , πηγαίνοντας από το πρώτο προς το τελευταίο, και σταματάει στο πρώτο που αντιστοιχεί στο URL που ζήτησε ο χρήστης. Στην περίπτωση που βρεθεί κάποιο ταιριαστό path, το django καλεί το αντίστοιχο view το οποίο ουσιαστικά είναι μια python συνάρτηση και περνά σε αυτό το request του χρήστη (object που περιέχει μεταδεδομένα σχετικά με το πώς ο χρήστης κάλεσε μια σελίδα) και τις τιμές που συλλέχθηκαν από το path. Για παράδειγμα, αν ένας χρήστης ζητήσει το URL : "/articles/2005/05/39323/" το Django θα καλέσει τη συνάρτηση:

```
news.views.article_detail(request, year=2005, month=5, pk=39323).
```

Αν δεν βρεθεί κανένα ταιριαστό path, τότε επιστρέφεται μια special-case 404 view.

3.2.4 Σχεδιασμός των Views

Όταν καλείται ένα view είναι υπεύθυνο δύο πράγματα. Ή να επιστρέψει ένα HttpResponse object το οποίο εμπεριέχει το περιεχόμενο για την σελίδα που ζητήθηκε σύμφωνα με το request , ή να επιστρέψει ένα Http error. Οι υπόλοιπες λειτουργίες καθορίζονται από εμάς. Σε γενικές γραμμές, ένα view ανακτά δεδομένα σύμφωνα με τις παραμέτρους του αντίστοιχου path , φορτώνει ένα Template και αποδίδει αυτό το template με τα δεδομένα που ανακτήθηκαν. Ένα παράδειγμα ενός view για το year_archive (βλ 3.2.3) θα μπορούσε να είναι το εξής:

```
from django.shortcuts import render

from .models import Article

def year_archive(request, year):
    a_list = Article.objects.filter(pub_date__year=year)
    context = {'year': year, 'article_list': a_list}
    return render(request, 'news/year_archive.html', context)
```

Εικόνα 31: Views. Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

3.2.5 Σχεδιασμός των Templates

Το Django περιέχει template search path στο οποίο μπορούμε να καθορίσουμε το όνομα των φακέλων στους οποίους θα ψάχνει για templates. Αν ένα Template δεν υπάρξει στον πρώτο φάκελο , τσεκάρει τον δεύτερο και ου το καθεξής.

Ο κώδικας του view year_archive(βλ 3.2.4) φορτώνει το template με όνομα year_archive.html. Έστω λοιπόν ότι το template αυτό υπάρχει και έχει τον εξής κώδικα:

```

{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
  <p>{{ article.headline }}</p>
  <p>By {{ article.reporter.full_name }}</p>
  <p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}

```

Εικόνα 32: Template example 1. Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

Οι μεταβλητές βρίσκονται ανάμεσα σε διπλά άγκιστρα. Για παράδειγμα το `{{ article.reporter.full_name }}` σημαίνει “Εμφάνισε την τιμή του `full_name` του `reporter` που σχετίζεται με το άρθρο `article` (στην συγκεκριμένη περίπτωση το `article` αναφέρεται στο `Article` object που έχει στο πεδίο `year` του `pub_date` την τιμή που κατοχύρωσε ο χρήστης στο URL. Το φιλτράρισμα του πίνακα `Article` γίνεται στην πρώτη γραμμή κώδικα του view (βλ 3.2.4).)”

Στην `{{ article.pub_date|date:"F j, Y" }}` χρησιμοποιείται ένα Unix-style “pipe” (το σύμβολο “|”). Ότι βρίσκεται μετά το σύμβολο αυτό ονομάζεται `template filter` και είναι ένας τρόπος να φιλτράρουμε την τιμή μιας μεταβλητής. Στην συγκεκριμένη περίπτωση το φίλτρο `date` διαμορφώνει ένα `datetime` object στην μορφή που του δίνουμε (“F j, Y”)

Επίσης το Django επιτρέπει το “template inheritance” δηλαδή να μπορεί ένα `template` να κληρονομήσει ένα άλλο. Στην περίπτωση μας αυτό κάνει η `{% extends "base.html" %}`. Πιο συγκεκριμένα, αυτό που κάνει αυτή η εντολή είναι να φορτώνει πρώτα το `template` με το όνομα `base` το οποίο έχει ορίσει “άδεια” τα οποία θα γεμίσουν τα `templates` που θα το καλέσουν.

Παράδειγμα ενός τέτοιου `template` είναι το παρακάτω:

```
{% load static %}
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    
    {% block content %}{% endblock %}
</body>
</html>
```

Εικόνα 33: Template example 2. Πηγή: <https://docs.djangoproject.com/en/2.2/intro/overview/>

ΚΕΦΑΛΑΙΟ 4

4.1 Φιλοσοφία και αρχικός σχεδιασμός της εφαρμογής

Όπως προαναφέρθηκε και στην εισαγωγή, σκοπός της εργασίας αυτής είναι η ανάπτυξη διαδικτυακής εφαρμογής η οποία έχει σκοπό την online δημιουργία μουσικών κοινωνιών.

Στην συγκεκριμένη υποενότητα θα αναλύσουμε την φιλοσοφία πίσω από αυτήν την εφαρμογή και θα δείξουμε τον αρχικό σχεδιασμό αυτής,. Ο τελευταίος είναι ουσιαστικά η περιγραφή της διάδρασης του χρήστη με την εφαρμογή μέσω περιγραφής των λειτουργιών της και εικόνων ,ώστε να γίνουν πιο ξεκάθαρες οι ανάγκες της εφαρμογής και να καθοριστεί ο τρόπος υλοποίησής της. Να σημειωθεί ότι ο αρχικός σχεδιασμός δεν συμπίπτει πλήρως με το τελικό αποτέλεσμα διότι κάποιες λειτουργίες που περιγράφονται σε αυτόν δεν πραγματοποιήθηκαν λόγω τεχνικών δυσκολιών ή προσεκτήθηκαν άλλες η οποίες εμπλουτίζουν καλύτερα την εφαρμογή.

4.1.2 Φιλοσοφία

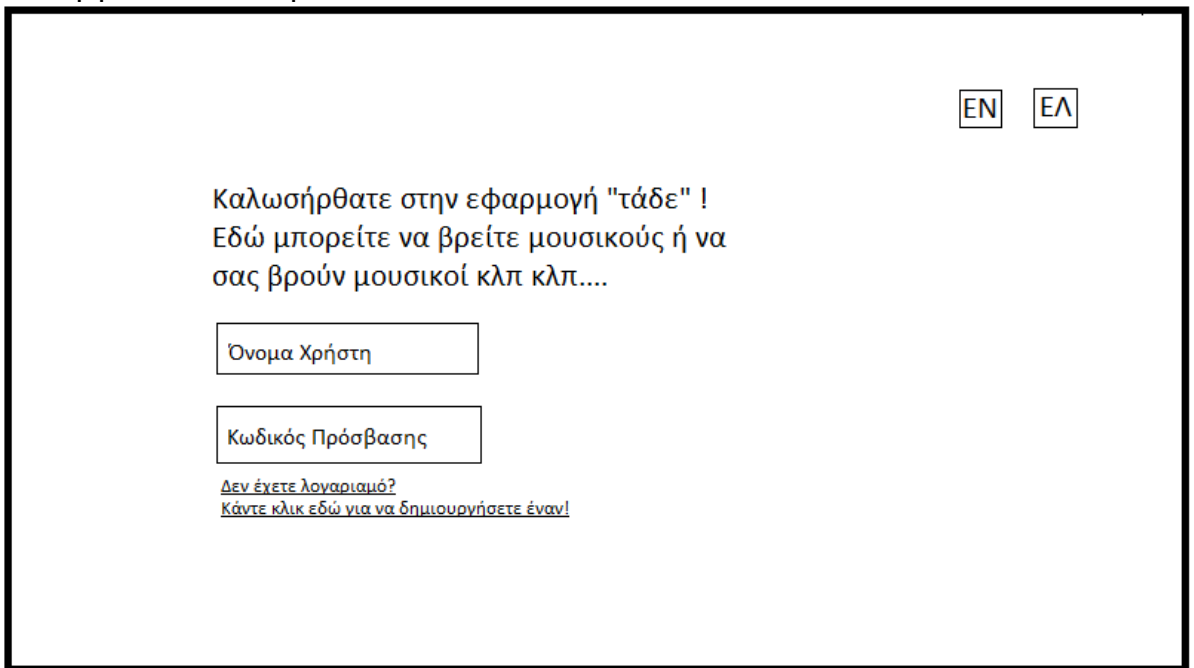
Όπως έχει αναφερθεί και στην εισαγωγή, σκοπός της εφαρμογής αυτή είναι να αποτελέσει εργαλείο μέσω του οποίου θα μπορεί ένας μουσικός να βρει άλλους μουσικούς με παρόμοια μουσικά ενδιαφέροντα ή να τον βρίσκουν άλλοι αντίστοιχα. Σκοπός αυτής της συνεύρεσης είναι η δημιουργία μπάντας από την αρχή, εύρεση μέλους για πάντα που υπάρχει έλλειψη (για παράδειγμα μια Post-punk που ψάχνει για drummer στα ίδια ακούσματα να μπορεί να τον βρει) ή για μια μουσική σύνθεση . Επίσης , θα πρέπει να υπάρχει δυνατότητα αξιολόγησης και επικοινωνίας μεταξύ των μουσικών.

Άρα αυτό που χρειαζόμαστε είναι ένας διαδικτυακός τόπος μέσω του οποίου ένας μουσικός θα μπορεί να φτιάξει ένα προφίλ το οποίο θα αποτελείται από προσωπικά στοιχεία, δηλαδή όνομα, επίθετο κλπ, και μουσικά στοιχεία τα οποία αποτελούνται από μουσικά ακούσματα(jazz, blues-rock, funk κλπ) και μουσικές ιδιότητες (κιθαρίστας , drummer κλπ). Μέσω αυτού του προφίλ θα μπορεί να ανεβάσει συνθέσεις και στίχους, να αναζητά μουσικούς, να τους αξιολογεί ακούγοντας συνθέσεις τους ή διαβάζοντας στίχους τους, και να μπορεί να στείλει μήνυμα σε κάποιον σε περίπτωση που τον ενδιαφέρει το μουσικό του προφίλ.

4.1.2 Αρχικός σχεδιασμός της εφαρμογής

Παρακάτω φαίνεται ο αρχικός σχεδιασμός της εφαρμογής όπου περιγράφονται λειτουργίες αυτής μαζί με παραδείγματα εικόνων

1. Ο χρήστης ανοίγει την εφαρμογή και βλέπει το login_page και το μήνυμα καλωσορίσματος μαζί με μια σύντομη περιγραφή της εφαρμογής. Εάν έχει λογαριασμό στην εφαρμογή συμπληρώνει τα πεδία όνομα χρήστη και κωδικός πρόσβασης, αλλιώς πρέπει να δημιουργήσει έναν λογαριασμό. Πάνω δεξιά στο login_page μπορεί να επιλέξει ανάμεσα σε Ελληνικά και Αγγλικά. Η default επιλογή είναι τα Ελληνικά



The screenshot shows a login page with a white background and a black border. In the top right corner, there are two buttons labeled 'EN' and 'EL'. The main text in the center reads: 'Καλωσήρθατε στην εφαρμογή "τάδε" !', 'Εδώ μπορείτε να βρείτε μουσικούς ή να', 'σας βρούν μουσικοί κλπ κλπ....'. Below this text are two input fields: the first is labeled 'Όνομα Χρήστη' and the second is labeled 'Κωδικός Πρόσβασης'. At the bottom, there is a link that says 'Δεν έχετε λογαριασμό?' followed by 'Κάντε κλικ εδώ για να δημιουργήσετε έναν!'.

Εικόνα 34: Login_page

2. Πατώντας το link για τη δημιουργία λογαριασμού (βλ login_page) κατευθύνεται στην registration_page_1 που αφορά την εισαγωγή των εξής στοιχείων: όνομα, επίθετο, email ή τηλέφωνο, όνομα χρήστη, κωδικός πρόσβασης, ημερομηνία γέννησης, φωτογραφία προφίλ. Όλα τα παραπάνω πεδία είναι υποχρεωτικά. Αφού συμπληρωθούν, μπορεί να πατήσει στο link «Επόμενο»

Όνομα

Επίθετο

Mail ή τηλέφωνο

Όνομα χρήστη

Κωδικός Πρόσβασης

Ημερομηνία γέννησης

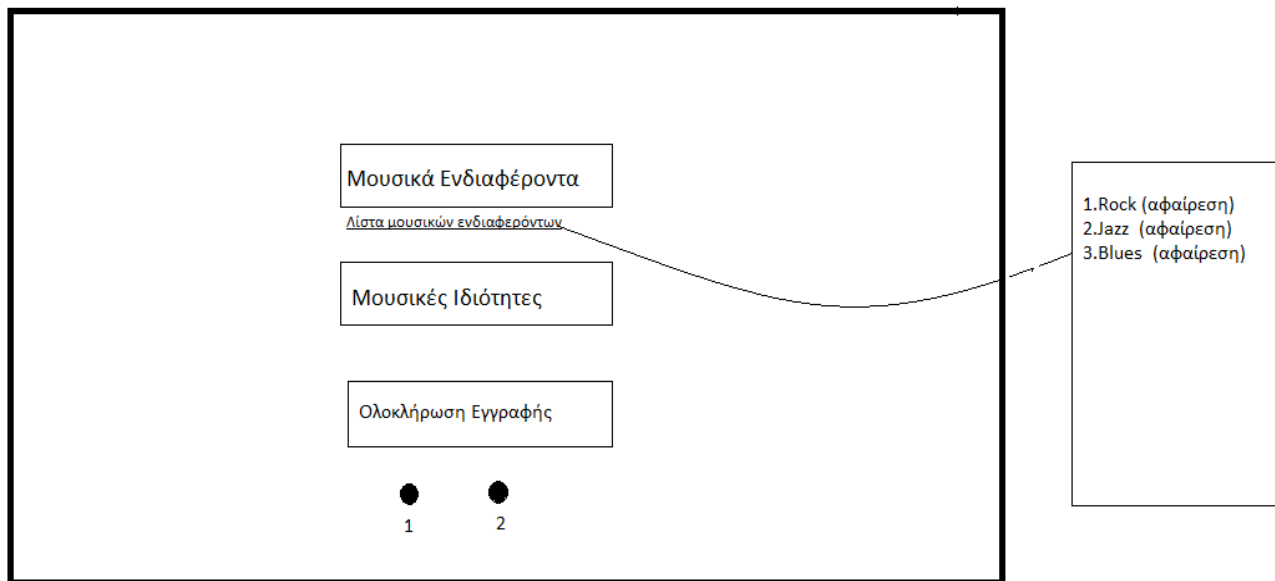
Επόμενο →

1 2

Κάντε κλικ εδώ για να εισάγετε την φωτογραφία προφίλ σας

Εικόνα 35: Registration_page_1

3. Πατώντας στο link «Επόμενο» (βλ registration_page_1) ο χρήστης κατευθύνεται στην registration_page_2 που αφορά την εισαγωγή των στοιχείων που αφορούν το μουσικό του προφίλ. Τα στοιχεία αυτά είναι: μουσικά ενδιαφέροντα(classic-rock, jazz κλπ) και μουσικές ιδιότητες (drummer, μπασίστας κλπ). Κάθε φορά που ο χρήστης πατά κάποιο γράμμα πάνω σε ένα πεδίο για να το συμπληρώσει, το autocomplete θα εμφανίζει προτάσεις (πχ αν στο πεδίο μουσικά ενδιαφέροντα πατήσει «P» τότε το autocomplete θα εμφανίσει :Post-Rock, Post-Punk κλπ. Αντίστοιχα και στις μουσικές ιδιότητες . Αν πατήσει «κ» εμφανίζεται κιθαρίστας.) Για να εισάγει το στοιχείο που θέλει, συμπληρώνει το αντίστοιχο πεδίο και πατά enter. Ύστερα, κάτω από το εκάστοτε πεδίο θα εμφανίζεται link που οδηγεί σε ξεχωριστή σελίδα όπου ο χρήστης θα μπορεί να βλέπει τι στοιχεία έχει εισαγάγει και να κάνει διορθώσεις. Ύστερα πατά ολοκλήρωση εγγραφής
- Να σημειωθεί ότι δεν μπορούν να εισαχθούν στοιχεία που δεν εμφανίζει το autocomplete. Κάτι τέτοιο θα προκαλούσε αποθήκευση άχρηστης πληροφορίας που θα έκανε πιο δύσκολη την συνεύρεση των μουσικών.



Εικόνα 36: Registration_page_2

4. Μετά την ολοκλήρωση της εγγραφής ο χρήστης κατευθύνεται στην home_page. Πάνω αριστερά βλέπει την φωτογραφία προφίλ του στην οποία αν κάνει κλικ οδηγείται πάλι στην home_page. Κάτω από την φωτογραφία προφίλ υπάρχουν links που το κάθε ένα οδηγεί σε κάποιο page. Επίσης, κάτω ή δίπλα από κάθε Link φαίνεται η ποσότητα των δεδομένων που αντιστοιχεί στο κάθε ένα (είναι όλα 0 διότι το προφίλ είναι καινούργιο).
- Μέσω του home_page ο χρήστης μπορεί να βρει μουσικό συμπληρώνοντας τουλάχιστον ένα από τα παρακάτω πεδία με την ίδια λογική όπως στο registration_page_2 (βλ registration_page_2) εκτός από το πεδίο όνομα χρήστη που το συμπληρώνει μόνο μία φορά. Αφού λοιπόν συμπληρώσει τουλάχιστον ένα από τα παραπάνω πεδία μπορεί να πατήσει «Αναζήτηση» ώστε να του εμφανιστούν μουσικοί που έχουν τα χαρακτηριστικά που έχει εισάγει (βλ research_page). Επίσης μέσω του link ανέβασμα αρχείου οδηγείται στην upload_page όπου μπορεί να ανεβάσει συνθέσεις ή στίχους (βλ upload_page).

Φωτο
Προφίλ

Συνδεδεμένοι μουσικοί (0)

Στίχοι (0)

Μ.Ο κριτικών (0)

Συνθέσεις (0)

Μ.Ο κριτικών (0)

Μηνύματα (0)

Αιτήματα συνδέσεων (0)

Ανέβασμα αρχείου

Συνδεθείτε με μουσικούς! Συμπληρώστε τουλάχιστον 1 από τα παρακάτω πεδία και βρείτε τον μουσικό που θέλετε

Μουσική ιδιότητα

Μουσικά ενδιαφέροντα

Όνομα χρήστη

Αναζήτηση

Εικόνα 37: Home_page

Παρακάτω γίνεται περιγραφή των Links που βρίσκονται κάτω από την φωτογραφία προφίλ.

Συνδεδεμένοι Μουσικοί: Λίστα με το όνομα χρήστη του κάθε μουσικού με τον οποίο ο χρήστης έχει συνδεθεί

Στίχοι: Λίστα με στίχους που έχει ανεβάσει ο χρήστης

Συνθέσεις: Λίστα με συνθέσεις που έχει ανεβάσει ο χρήστης. Επίσης , δίπλα από τις συνθέσεις και τους στίχους υπάρχει και ο μέσος όρος των κριτικών που είναι ουσιαστικά ο μέσος όρος των βαθμολογιών όλων των στίχων ή συνθέσεων

Μηνύματα: Λίστα με συνομιλίες με άλλους χρήστες

Αιτήματα συνδέσεων: Λίστα με αιτήματα συνδέσεων από άλλους χρήστες

- Κάνοντας κλικ στο «Ανέβασμα αρχείου» ο χρήστης κατευθύνεται στην upload_page. Εδώ μπορεί να ανεβάσει αρχείο το οποίο θα ανήκει σε μία από τις 2 κατηγορίες (στίχοι, σύνθεση).Κάτω από κάθε κατηγορία, υπάρχει περιγραφή για τι τύπου αρχείου αντιστοιχεί σε αυτήν(pdf, mp3 κλπ) και ένα πεδίο το οποίο συμπληρώνεται με την ίδια λογική όπως το registration_page και αφορά την κατηγοριοποίηση του αρχείου σε κάποιο μουσικό είδος. Πατώντας στο link «Ανέβασμα αρχείου» το αρχείο ανεβαίνει και προστίθεται στη λίστα Στίχοι ή Σύνθεση(βλ home_page) αναλόγως σε ποια κατηγορία ανήκει

Στίχοι (pdf, doc κλπ) Μουσικό είδος Ανέβασμα αρχείου	Σύνθεση (mp3, mp4, flac κλπ) Μουσικό είδος Ανέβασμα αρχείου
--	---

Εικόνα 38: Upload_page

6. Κάνοντας κλικ στο link «Αναζήτηση» (βλ home_page) ο χρήστης βλέπει λίστα με μουσικού που οι μουσικές τους ιδιότητες, τα μουσικά ακούσματα ή το όνομα χρήστη αντιστοιχούν στα στοιχεία που εισήγαγε στα αντίστοιχα πεδία στο home_page. Αν ο χρήστης κάνει κλικ σε κάποιο Link που αντιστοιχεί σε κάποιον χρήστη, μεταφέρεται στο προφίλ του όπου βλέπει όλες τις μουσικές ιδιότητες, τα μουσικά ακούσματα καθώς και αρχεία που έχει ανεβάσει και κριτικές (αν έχουν).

	Όν. χρήστη : p14pano	
	Αίτημα σύδεσης Μήνυμα	
	Όν. χρήστη : musicLover	
	Αίτημα σύνδεσης Μήνυμα	

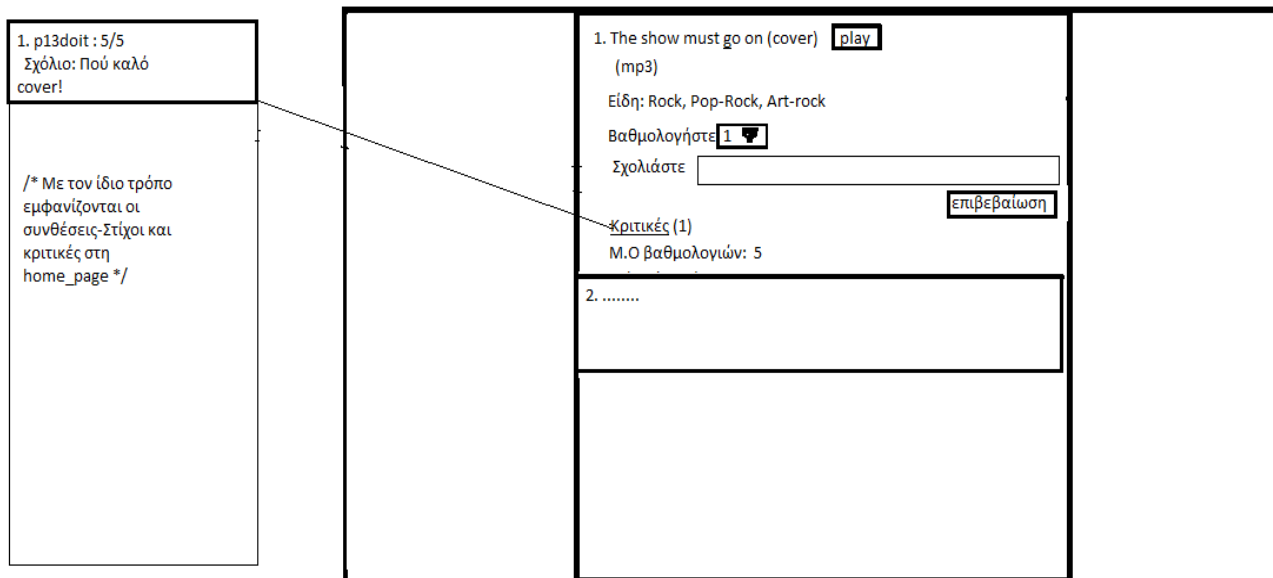
Εικόνα 39: Research_page

7. Έστω ότι ο χρήστης έκανε κλικ στο προφίλ musicLover. Τότε κατευθύνεται στην profile_page του musicLover όπου βλέπει αναλυτικά τις μουσικές ιδιότητες και τα μουσικά ακούσματα καθώς και την φωτογραφία προφίλ του. Για να δει και τα προσωπικά του στοιχεία θα πρέπει να του κάνει αίτημα σύνδεσης και αυτό να γίνει δεκτό. Μπορεί επίσης μέσω του Link «Μήνυμα» να του στείλει μήνυμα και μέσω των Links «Στίχοι» , «Συνθέσεις» και «Συνδεδεμένοι μουσικοί» να δει την λίστα με τις συνθέσεις , τους στίχους του ή τους μουσικούς με τους οποίους είναι συνδεδεμένος.

Εικόνα Προφίλ	Μήνυμα	Αίτημα σύνδεσης
Όνομα χρήστη : musicLover Μουσικές ιδιότητες : Drummer Μουσικά ενδιαφέροντα : Rock, Blues,.....κλπ Προσωπικά στοιχεία: (Για να φανούν τα προσωπικά στοιχεία πρέπει ο χρήστης musicLover να δεχθεί το αίτημα σύνδεσής σας) Στίχοι(1) Μ.Ο κριτικών 4/5 Συνθέσεις(2) Μ.Ο κριτικών 5/5 Συνδεδεμένοι μουσικοί(2)		

Εικόνα 39: Profile_page

8. Ο χρήστης επιλέγει να δει την λίστα με τις συνθέσεις. Για κάθε σύνθεση βλέπει τον τίτλο της, τον τύπο αρχείου, τα είδη στα οποία ανήκει και τον μέσο όρο των βαθμολογιών. Επίσης μπορεί να δει και τις κριτικές που υπάρχουν οι οποίες αποτελούνται από βαθμολογία και σχόλιο αιτιολόγησης της βαθμολογίας. Ο χρήστης μπορεί επίσης να βαθμολογήσει από το 1 έως το 5 και να σχολιάσει την βαθμολογία αυτή προαιρετικά, Για την επικύρωση της κριτικής του πατά «επιβεβαίωση»



Εικόνα 40: Composition_list_page

4.2 Τρόπος εγκατάστασης και περιγραφή των λειτουργιών του project σε αρχάριο επίπεδο

Σε αυτή την υποενότητα θα δείξουμε γιατί επιλέχθηκε το Django για την δημιουργία του συγκεκριμένου project, τον τρόπο με τον οποίο το εγκαταστήσαμε καθώς και τη προκαθορισμένη λειτουργικότητα που μας παρέχει μόλις δημιουργήσουμε το project μας.

4.2.1 Επιλογή εργαλείου για την υλοποίηση της εφαρμογής

Έχοντας υπόψιν τον παραπάνω σχεδιασμό της εφαρμογής (βλ 4.1.2) προτιμήθηκε για λόγους ευχρηστίας ,το εργαλείο που θα χρησιμοποιηθεί για την υλοποίησή της να είναι κάποιο web framework(χρησιμότητα και οι ιδιότητες ενός web framework αναλύονται στο 1^ο κεφάλαιο). Έχοντας λοιπόν ως πιο χρησιμοποιούμενη γλώσσα προγραμματισμού την python, αναζητήσα web frameworks της python. Μέσα από αυτήν την αναζήτηση και λαμβάνοντας υπόψιν κριτικές άλλων προγραμματιστών για διάφορα frameworks της python που υπάρχουν κατέληξα στο Django. Πιο συγκριμένοι λόγοι για την επιλογή του Django είναι ότι παρέχει το δικό του authentication system που είναι πολύ χρήσιμο για την εφαρμογή μου, παρέχει επαναχρησιμοποιούμενο κώδικα , secure system,δικό του web server, και στηρίζεται στο μοντέλο MVC(βλ 2.2.2) που παρέχει πρακτικότητα στο να έχουμε τον έλεγχο των λειτουργιών της εφαρμογής σου.

4.2.2 Εγκατάσταση του Django για την υλοποίηση της εφαρμογής

Όπως έχει προαναφερθεί, το πρώτο βήμα για την εγκατάσταση του Django είναι η εγκατάσταση της python (βλ 3.1.1). έχοντας ήδη εγκατεστημένη την python 3.6.0 μπορούσα να προβώ απευθείας στην εγκατάσταση του Django. Ο τρόπος που επέλεξα να το κάνω ήταν μέσω του εργαλείου `easy_install`, το οποίο εγκαθίσταται αυτόματα μαζί με την python και χρησιμεύει στο να κατεβάζουμε, να εγκαθιστούμε και να διαχειριζόμαστε python packages αυτόματα. Για να επιβεβαιώσω την ορθή λειτουργία του `easy_install` έτρεξα στο command line ως administrator την εντολή

```
easy_install -version
```

που δείχνει την έκδοση που είναι εγκατεστημένη στον υπολογιστή μας:

```
C:\Users\user>easy_install --version  
setuptools 28.8.0 from c:\users\user\appdata\local\programs\python\python36-32\lib\site-packages (Python 3.6)
```

Εικόνα 41: Easy_install

Εφόσον εμφανίστηκε η έκδοση του `easy_install` σημαίνει ότι λειτουργεί σωστά. Με την εντολή `easy_install django` εγκατέστησα την τελευταία έκδοση του Django την συγκεκριμένη χρονική περίοδο. Για να επιβεβαιώσω την ορθή εγκατάστασή του έτρεξα την εντολή

```
django-admin -version
```

Η έκδοση του Django εμφανίστηκε άρα η εγκατάσταση ήταν επιτυχής:

```
C:\Users\user>django-admin --version  
2.1.4
```

Εικόνα 42: Django-admin version

4.2.3 Δημιουργία του project και του app

4.2.3.1 Δημιουργία του project

Για να δημιουργήσω ένα νέο project με όνομα `musician_finder` έτρεξα την εντολή

```
django-admin startproject musician_finder
```

επιλέγοντας, μέσω του command line, τον φάκελο desktop ως μέρος εγκατάστασης του project. Το project που δημιουργήθηκε από την παραπάνω εντολή αποτελεί φάκελο ο οποίος περιέχει υποφάκελο με το όνομα του project μας και ένα python αρχείο που ονομάζεται manage.py. Μετά άνοιξα το project με το IDE Pycharm με την έκδοση 2018.3.2 x64. Να σημειωθεί ότι μέσω του συγκεκριμένου IDE επεξεργαζόμουν την εφαρμογή μου.

Στον υποφάκελο που προαναφέρθηκε περιέχονται τα εξής python αρχεία όπου γίνεται και επεξήγηση του πώς λειτουργεί το κάθε ένα:

1. **_init_.py** : Αρχείο που «λέει» στην python να αντιμετωπίζει τον φάκελο στον οποίο βρίσκεται ως python package. Δεν περιέχει καθόλου κώδικα
2. **settings.py**: Αποτελεί την κεντρική διαμόρφωση όλου του project. Πριν κάνουμε κάποια προσθήκη στον κώδικά του εμπεριέχονται τα εξής:

Αρχικά βλέπουμε σχόλια που μας δίνουν πληροφορίες για την έκδοση του Django καθώς και sites που δίνουν οδηγίες για το πώς δουλεύουν οι εντολές μέσα σε αυτό το αρχείο καθώς και την λίστα όλων των εντολών και τις τιμές αυτών:

```
"""
Django settings for musician_finder project.

Generated by 'django-admin startproject' using Django 2.1.4.

For more information on this file, see
https://docs.djangoproject.com/en/2.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.1/ref/settings/
"""
```

Απόσπασμα 1 : Αρχικά σχόλια στο settings.py

Παρακάτω βλέπουμε την εντολή :

```
import os
```

Απόσπασμα 2 : Μεταβλητή os

Το os είναι python package που χρησιμοποιείται για την πρόσβαση σε paths του υπολογιστή μας. Παράδειγμα της χρήσης αυτού του package είναι η εντολή που βλέπουμε παρακάτω που καθορίζει το πού βρίσκεται ο base directory .

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

Απόσπασμα 3 : BASE_DIR path

Παρακάτω βλέπουμε το SECRET_KEY (για λόγους ασφάλειας δεν αποκαλύπτεται). Χρησιμότητα αυτού είναι να λειτουργεί ως input σε cryptographic hash function (μαθηματική συνάρτηση που έχοντας ως είσοδο μια αυθαίρετου μεγέθους ομάδα δεδομένων δίνει έξοδο μια καθορισμένου μεγέθους συμβολοσειρά) για τα δεδομένα της εφαρμογής μας με σκοπό την ασφάλεια

Παρακάτω βλέπουμε την εντολή:

```
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True
```

Απόσπασμα 4 : DEBUG

Η μεταβλητή DEBUG ,όταν έχει την τιμή true, εξετάζει αν υπάρχει κάποιο error στον κώδικα που εκτελούμε και μας δίνει λεπτομερή επεξήγηση για το error αυτό καθώς και σε ποια γραμμή του κώδικα βρίσκεται. Επειδή όμως αποκαλύπτει πολλές πληροφορίες για την εφαρμογή μας τότε, για λόγους ασφάλειας, όταν το site μας εκτελείται και από άλλους χρήστες πρέπει να το απενεργοποιήσουμε δίνοντάς του την τιμή false.

Μετά βλέπουμε την λίστα ALLOWED_HOSTS. Σε αυτήν εκχωρούνται domain names που μπορεί να εξυπηρετήσει το site μας. Εάν η λίστα είναι άδεια τότε το domain name είναι <http://127.0.0.1:8000/>:

```
ALLOWED_HOSTS = []
```

Απόσπασμα 5 : ALLOWED_HOSTS

Μετά βλέπουμε την λίστα INSTALLED_APPS που περιέχει τα εγκατεστημένα apps της διαδικτυακής εφαρμογής μας:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

Απόσπασμα 6 : INSTALLED_APPS

'django.contrib.admin' : Framework μέσω του οποίου επεξεργαζόμαστε το backend χωρίς κώδικα. Σε επόμενη υποενότητα θα δείξουμε πως λειτουργεί

'django.contrib.auth', : Περιέχει τον πυρήνα του authentication framework του Django και τα προεπιλεγμένα μοντέλα που σχετίζονται με το framework αυτό

'django.contrib.contenttypes', : Framework που παρέχει υψηλού επιπέδου interface για να επεξεργαζόμαστε τα μοντέλα της βάσης μας.

'django.contrib.sessions', : Framework που επιτρέπει την δημιουργία συνόδων για τον κάθε χρήστη του site

'django.contrib.messages', : Χειρίζεται μηνύματα που ενημερώνουν τον χρήστη (πχ error messages)

'django.contrib.staticfiles', : Framework που χειρίζεται όλα τα static αρχεία της διαδικτυακής εφαρμογής μας (αρχεί τύπου html, css, javascript, jpg κλπ)

Μετά ακολουθεί η λίστα MIDDLEWARE. Η λίστα αυτή περιέχει μικρά packages λογισμικού που χειρίζονται την επικοινωνία μεταξύ του server και του framework μας.

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

Απόσπασμα 7 : MIDDLEWARE

Μετά βλέπουμε την εντολή:

```
ROOT_URLCONF = 'musician_finder.urls'
```

Απόσπασμα 8 : ROOT_CONF

Στην μεταβλητή ROOT_CONF εκχωρείται το path που οδηγεί στο αρχείο που θέλουμε να είναι root για τα url μας.

Παρακάτω βλέπουμε την λίστα TEMPLATES:

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    ],  
]
```

Απόσπασμα 9 : TEMPLATES

Η λίστα αυτή περιέχει configurations που καθορίζουν την λειτουργία του template system του Django. Ρόλος αυτού του template system είναι να παρέχεται στον χρήστη του Django εύκολος τρόπος εισαγωγής δυναμικού περιεχομένου στα templates.

Παρακάτω βλέπουμε την εντολή:

```
WSGI_APPLICATION = 'musician_finder.wsgi.application'
```

Απόσπασμα 10 : WSGI_APPLICATION

Στην παραπάνω μεταβλητή εκχωρείται το path που δείχνει την τοποθεσία της εφαρμογής wsgi που ρόλος της είναι να βοηθά την επικοινωνία μεταξύ του server και του project μας.

Μετά βλέπουμε το λεξικό DATABASES:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Απόσπασμα 11 : DATABASES

Τα παραπάνω λεξικό περιέχει τις ρυθμίσεις για όλες τις βάσεις δεδομένων που θα χρησιμοποιηθούν από το Django. Στο συγκεκριμένο project η βάση που χρησιμοποιείται είναι η SQLite δηλαδή η προκαθορισμένη βάση.

Παρακάτω βλέπουμε τη λίστα AUTH_PASSWORD_VALIDATORS:

```
# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
```

```
'django.contrib.auth.password_validation.NumericPasswordValidator',  
},  
]
```

Απόσπασμα 12 : AUTH_PASSWORD_VALIDATORS

Η λίστα αυτή περιέχει μικρά packages λογισμικού που επιτρέπουν την αυθεντικοποίηση των passwords.

Μετά βλέπουμε τις εξής μεταβλητές

```
LANGUAGE_CODE = 'en-us'  
  
TIME_ZONE = 'UTC'  
  
USE_I18N = True  
  
USE_L10N = True  
  
USE_TZ = True
```

Απόσπασμα 13: LANGUAGE, TIME_ZONE, USE

LANGUAGE_CODE: Η default γλώσσα του projects μας

TIME_ZONE: Η ζώνη ώρας του projects μας

USE_I18N : Boolean μεταβλητή που καθορίζει αν ενεργοποιείται το translation system του Django

USE_L10N: Boolean μεταβλητή που καθορίζει αν τα δεδομένα θα απεικονίζονται αναλόγως με την περιοχή που ανήκει στην ζώνη ώρας που καθορίστηκε στην TIME_ZONE

USE_TZ: Boolean μεταβλητή που καθορίζει αν χρησιμοποιηθεί εφαρμοστεί η ζώνη ώρας που καθορίστηκε στην TIME_ZONE

Τέλος βλέπουμε την εντολή:

```
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/2.1/howto/static-files/  
  
STATIC_URL = '/static/'
```

Απόσπασμα 14: STATIC_URL

Εδώ καθορίζεται το path στο οποίο θα αποθηκεύονται τα static αρχεία του project μας

3. **urls.py:** Μέσω αυτού του αρχείου καθορίζουμε τα paths του projects μας Ο τρόπος που γίνεται αυτό είναι συνδέοντας ένα url με ένα view. Τα views

αποτελούν συναρτήσεις που συνδέουν την μοντέλα της βάσης μας με templates. Σε επόμενη υποενότητα φαίνεται αναλυτικότερα η λειτουργικότητά τους μέσω παραδειγμάτων.

Αρχικά βλέπουμε σχόλια με οδηγίες για το πώς να δημιουργούμε paths:

```
"""musician_finder URL Configuration

The `urlpatterns` list routes URLs to views. For more information please
see:
    https://docs.djangoproject.com/en/2.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
```

Απόσπασμα 15: urls.py σχόλια

Ύστερα βλέπουμε τον εξής κώδικα:

```
from django.contrib import admin
from django.urls import path

urlpatterns = [

    path('admin/', admin.site.urls),

]
```

Απόσπασμα 15: urls.py κώδικας

Αυτό που κάνει ο παραπάνω κώδικας ο καθορισμός path που συνδέει το url 'admin/' με το view admin.site.urls

- 4. wsgi.py:** Βγαίνει από τα αρχικά Web server gateway interface που, όπως προαναφέρθηκε, ρόλος της είναι να βοηθά την επικοινωνία μεταξύ του server και του project μας.
- 5. manage.py:** Βοηθητικό πρόγραμμα που μέσω αυτού εκτελούμε administrative task στην command line

4.2.3.2 Δημιουργία του app

Για δημιουργήσω μια εφαρμογή για την διαδικτυακή εφαρμογή μου έτρεξα στο command line, εφόσον βρίσκομαι στον φάκελο του project μου, την εντολή

```
python manage.py startapp users_core
```

. Τότε δημιουργήθηκε υποφάκελος με όνομα **users_core** στον φάκελο του project μου και περιέχει τα εξής αρχεία:

1. **Migrations:** Φάκελος που περιέχει python αρχεία που σχετίζουν τον κώδικά μας με την βάση δεδομένων
2. **__init__.py** : Αρχείο που «λέει» στην python να αντιμετωπίζει τον φάκελο στον οποίο βρίσκεται ως Python package. Δεν περιέχει καθόλου κώδικα
3. **admin.py:** Αρχείο που παρέχει πλατφόρμα για να διαχειριζόμαστε την βάση μας .
4. **apps.py:** Αποτελεί configuration file για την εφαρμογή μας
5. **models.py:** Αρχείο που μέσω αυτού μπορούμε να φτιάξουμε τις αναπαραστάσεις των δεδομένων της εφαρμογής μας στη βάση
6. **tests.py:** Αρχείο που μέσω αυτού μπορούμε να κάνουμε δοκιμές στον κώδικά μας για bugs
7. **views.py:** Αρχείο που μέσω αυτού μπορούμε να φτιάξουμε συναρτήσεις που θα καλούνται από urls που έχουμε ορίσει στο **urls.py** (βλ 4.2.3.1) και επιστρέφουν ένα template.

4.3 Περιγραφή μοντέλων και λειτουργιών της τελειοποιημένης εφαρμογής

Στην υποενότητα αυτή γίνεται παρουσίαση των λειτουργιών της εφαρμογής στο τελικό της στάδιο. Η κάθε λειτουργία περιγράφεται με εικόνες που δείχνουν τι βλέπουμε κατά την περιήγησή μας στην εφαρμογή και με ανάλυση του κώδικα πίσω από αυτό που βλέπουμε. Επίσης , γίνεται και περιγραφή των μοντέλων που αναπαριστούν τα δεδομένα στη βάση δεδομένων.

Τα μοντέλα της βάσης μας είναι τα:

Profile: Αναπαριστά δεδομένα στη βάση που σχετίζονται με τα profile objects

Song: Αναπαριστά δεδομένα στη βάση που σχετίζονται με τις συνθέσεις

Lyrics: Αναπαριστά δεδομένα στη βάση που σχετίζονται με τους στίχους

Παρακάτω βλέπουμε τον κώδικα για το κάθε μοντέλο στο **models.py**:

Profile:

```
class Profile(models.Model):
    ROCK_CHOICES = (
        ('Classic Rock', 'Classic Rock'),
        ('Progressive Rock', 'Progressive Rock'),
        ('Post Rock', 'Post Rock'),
        ('Rock&Roll', 'Rock&Roll'),
        ('Psychedelic Rock', 'Psychedelic Rock'),
        ('Hard Rock', 'Hard Rock'),
        ('Garage Rock', 'Garage Rock'),
        ('Surf Rock', 'Surf Rock'),
        ('Rockabilly', 'Rockabilly'),
    )
    METAL_CHOICES = (
        ('Black Metal', 'Black Metal'),
        ('Death Metal', 'Death Metal'),
        ('Thrash Metal', 'Thrash Metal'),
        ('Doom Metal', 'Doom Metal'),
        ('Progressive Metal', 'Progressive Metal'),
        ('Sludge Metal', 'Sludge Metal'),
        ('Heavy Metal', 'Heavy Metal'),
        ('Metalcore', 'Metalcore'),
        ('Grindcore', 'Grindcore'),
    )
    PUNK_CHOICES = (
        ('Punk Rock', 'Punk Rock'),
        ('Pop Punk', 'Pop Punk'),
        ('Garage Punk', 'Garage Punk'),
        ('Hardcore Punk', 'Hardcore Punk'),
        ('Thrash Punk', 'Thrash Punk'),
        ('Crust Punk', 'Crust Punk'),
        ('Folk Punk', 'Folk Punk'),
        ('Ska Punk', 'Ska Punk'),
    )
    POP_CHOICES = (
        ('Indie Pop', 'Indie Pop'),
        ('Synth Pop', 'Synth Pop'),
        ('Power Pop', 'Power Pop'),
        ('Dream Pop', 'Dream Pop'),
        ('Noise Pop', 'Noise Pop'),
        ('Experimental Pop', 'Experimental Pop'),
        ('Jangle Pop', 'Jangle Pop'),
    )
    BLUES_CHOICES = (
        ('Blues', 'Blues'),
        ('Blues Rock', 'Blues Rock'),
        ('Country Blues', 'Country Blues'),
        ('Rythm & Blues', 'Rythm & Blues'),
        ('Electric Blues', 'Electric Blues'),
        ('Boogie-Woogie', 'Boogie-Woogie'),
        ('Bluegrass', 'Bluegrass'),
        ('Delta Blues', 'Delta Blues'),
    )
    JAZZ_CHOICES = (
        ('Jazz', 'Jazz'),
        ('Nu Jazz', 'Nu Jazz'),
        ('Modern Jazz', 'Modern Jazz'),
        ('Free Jazz', 'Free Jazz'),
```

```

        ('Soul Jazz', 'Soul Jazz'),
        ('Swing', 'Swing'),
        ('Latin Jazz', 'Latin Jazz'),
        ('Vocal Jazz', 'Vocal Jazz'),
        ('Spiritual Jazz', 'Spiritual Jazz'),
    )
    FUNK_CHOICES = (
        ('Funk', 'Funk'),
        ('Funk Jam', 'Funk Jam'),
        ('Deep Funk', 'Deep Funk'),
        ('Funk Rock', 'Funk Rock'),
        ('Jazz Funk', 'Jazz Funk'),
        ('g-Funk', 'g-Funk'),
        ('Electro Funk', 'Electro Funk'),
        ('Go-go', 'Go-go'),
    )
    SOUL_CHOICES = (
        ('Soul', 'Soul'),
        ('Neo-Soul', 'Neo-Soul'),
        ('Gospel', 'Gospel'),
    )
    ELECTRONIC_CHOICES = (
        ('House', 'House'),
        ('Electronica', 'Electronica'),
        ('Downtempo', 'Downtempo'),
        ('Techno', 'Techno'),
        ('Electro', 'Electro'),
        ('Dubstep', 'Dubstep'),
        ('Drum & Bass', 'Drum & Bass'),
        ('Trance', 'Trance'),
        ('Dub', 'Dub'),
        ('Vaporwave', 'Vaporwave'),
    )
    HIP_HOP_RAP_CHOICES = (
        ('Rap', 'Rap'),
        ('Trap', 'Trap'),
        ('Instrumental Hip-Hop', 'Instrumental Hip-Hop'),
        ('Boom-Bap', 'Boom-Bap'),
        ('Beat-tape', 'Beat-tape'),
    )
    GREEK_CHOICES = (
        ('Έντεχνο', 'Έντεχνο'),
        ('Λαϊκό', 'Λαϊκό'),
        ('Ρεμπέτεκο', 'Ρεμπέτεκο'),
    )
    OTHER_CHOICES = (
        ('Ska', 'Ska'),
        ('No Wave', 'No Wave'),
        ('Folk', 'Folk'),
        ('New wave', 'New wave'),
        ('Swing', 'Swing'),
        ('Gospel', 'Gospel'),
    )
    MUSIC_NATURE = (
        ('Drummer', 'Drummer'),
        ('Τραγουδιστής', 'Τραγουδιστής'),
        ('Κιθαρίστας', 'Κιθαρίστας'),
        ('Μπασίστας', 'Μπασίστας'),
    )

```

```

        ('Στιχουργός', 'Στιχουργός '),
        ('Σαξοφωνίστας', 'Σαξοφωνίστας '),
        ('Κλαριτζής', 'Κλαριτζής '),
        ('Dj', 'Dj '),
    )
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg',
upload_to='profile_pics')
    rock = MultiSelectField(choices=ROCK_CHOICES, blank=True)
    metal = MultiSelectField(choices=METAL_CHOICES, blank=True)
    punk = MultiSelectField(choices=PUNK_CHOICES, blank=True)
    pop = MultiSelectField(choices=POP_CHOICES, blank=True)
    blues = MultiSelectField(choices=BLUES_CHOICES, blank=True)
    jazz = MultiSelectField(choices=JAZZ_CHOICES, blank=True)
    funk = MultiSelectField(choices=FUNK_CHOICES, blank=True)
    soul = MultiSelectField(choices=SOUL_CHOICES, blank=True)
    electronic = MultiSelectField(choices=ELECTRONIC_CHOICES, blank=True)
    hiphop_rap = MultiSelectField(choices=HIP_HOP_RAP_CHOICES, blank=True)
    greek = MultiSelectField(choices=GREEK_CHOICES, blank=True)
    other = MultiSelectField(choices=OTHER_CHOICES, blank=True)
    music_nature = MultiSelectField(choices=MUSIC_NATURE, blank=True)
    birth_date = models.DateField(blank=True, null=True)

    def __str__(self):
        return f'{self.user.username} Profile'

    def save(self, *args, **kwargs):
        super(Profile, self).save(*args, **kwargs)

        img = Image.open(self.image.path)

        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size)
            img.save(self.image.path)

```

Απόσπασμα περιγραφής μοντέλων 1: Profile

Με το πεδίο **user** δημιουργείται μια σχέση ένας προς έναν μεταξύ των profile objects και των user objects και ορίζεται ότι αν διαγραφεί ένα user object τότε αυτομάτως διαγράφεται και το profile object που σχετίζεται με αυτό.

Το πεδίο **image** αναπαριστά την εικόνα προφίλ στη βάση. Επίσης έχει προκαθορισμένη εικόνα όταν δημιουργείται ένα νέο προφίλ και ορίζει ότι εάν αναρτηθεί άλλη εικόνα θα αποθηκεύεται στον φάκελο “profile pics” που είναι υποφάκελος του φακέλου που έχουμε ορίσει στο **settings.py** να αποθηκεύονται τα αρχεία που ανεβάζει ο χρήστης (βλ 4.3.5).

Τα πεδία **rock** έως **music_nature** αναπαριστούν τις πολλαπλές επιλογές για το κάθε μουσικό είδος οι οποίες έχουν οριστεί παραπάνω

Το πεδίο **birth_date** αναπαριστά την ημερομηνία γεννήσεως του προφίλ στη βάση

Η συνάρτηση **__str__()** καθορίζει ένα profile object να αναπαρίσταται ως “Όνομα χρήστη Profile”. Για παράδειγμα αντί για profile object(1) να εμφανίζεται admin Profile

Η συνάρτηση **save()** λαμβάνει την εικόνα που αναρτά ο χρήστης στο προφίλ του και εξετάζει αν το ύψος ή το πλάτος της ξεπερνά τα 300 pixels. Αν ναι τότε τροποποιεί το μέγεθος της εικόνας σε 300x300.

Song:

```
class Song(models.Model):
    composition_title = models.CharField(max_length=100, blank=True)
    composition = models.FileField(upload_to='uploaded_music', blank=True,
    validators=[FileExtensionValidator(allowed_extensions=['mp3', 'flac',
    'wav'])])
    user = models.ForeignKey(User, on_delete=models.CASCADE, default=None)
    is_public = models.BooleanField(default=False)

    def __str__(self):
        return f'{self.composition_title} composition'
```

Απόσπασμα περιγραφής μοντέλων 2: Song

Το πεδίο **composition_title** αναπαριστά τον τίτλο της σύνθεσης στη βάση. Ορίζεται να επιτρέπονται 100 χαρακτήρες.

Το πεδίο **composition** αναπαριστά το αρχείο της σύνθεσης στη βάση. Ορίζεται να αποθηκεύεται στον φάκελο “uploaded music” το αρχείο της σύνθεσης που αναρτάται και να πρέπει να έχει κατάληξη ανάμεσα σε mp3, flac, wav

Το πεδίο **user** σχετίζει συνθέσεις με συγκεκριμένο χρήστη. Ορίζεται ότι εάν διαγραφεί ο χρήστης διαγράφονται αυτομάτως και οι συνθέσεις που σχετίζονται.

Το πεδίο **is_public** όταν είναι True τότε η σύνθεση φαίνεται στους άλλους χρήστες ενώ όταν είναι False δεν φαίνεται. Ορίζεται ως προκαθορισμένη τιμή η False

Η συνάρτηση **__str__()** καθορίζει ένα song object να αναπαρίσταται ως ‘Τίτλος σύνθεσης composition’.

Lyrics:

```
class Lyrics(models.Model):
    lyrics_title = models.CharField(max_length=100, blank=True)
    lyrics = models.FileField(upload_to='uploaded_lyrics', blank=True,
    validators=[FileExtensionValidator(allowed_extensions=['pdf'])])
    user = models.ForeignKey(User, on_delete=models.CASCADE, default=None)

    def __str__(self):
        return f'{self.lyrics_title} lyrics'
```

Απόσπασμα περιγραφής μοντέλων 3: Lyrics

Τα πεδία του παραπάνω μοντέλου έχουν την ίδια λειτουργικότητα με τα προαναφερθέντα πεδία του μοντέλου **song** απλά χρησιμοποιείται για στίχους αντί για σύνθεσης. Οι διαφορές είναι στο ότι ένα αρχείο στίχων πρέπει να έχει την κατάληξη

pdf, τα αρχεία αποθηκεύονται στον φάκελο uploaded_lyrics και ένα lyrics object να αναπαρίσταται ως 'Τίτλος στίχων lyrics'.

Για την περιγραφή της εφαρμογής, αφού ενεργοποιηθεί ο server, ο χρήστης ακολουθεί τα εξής βήματα:

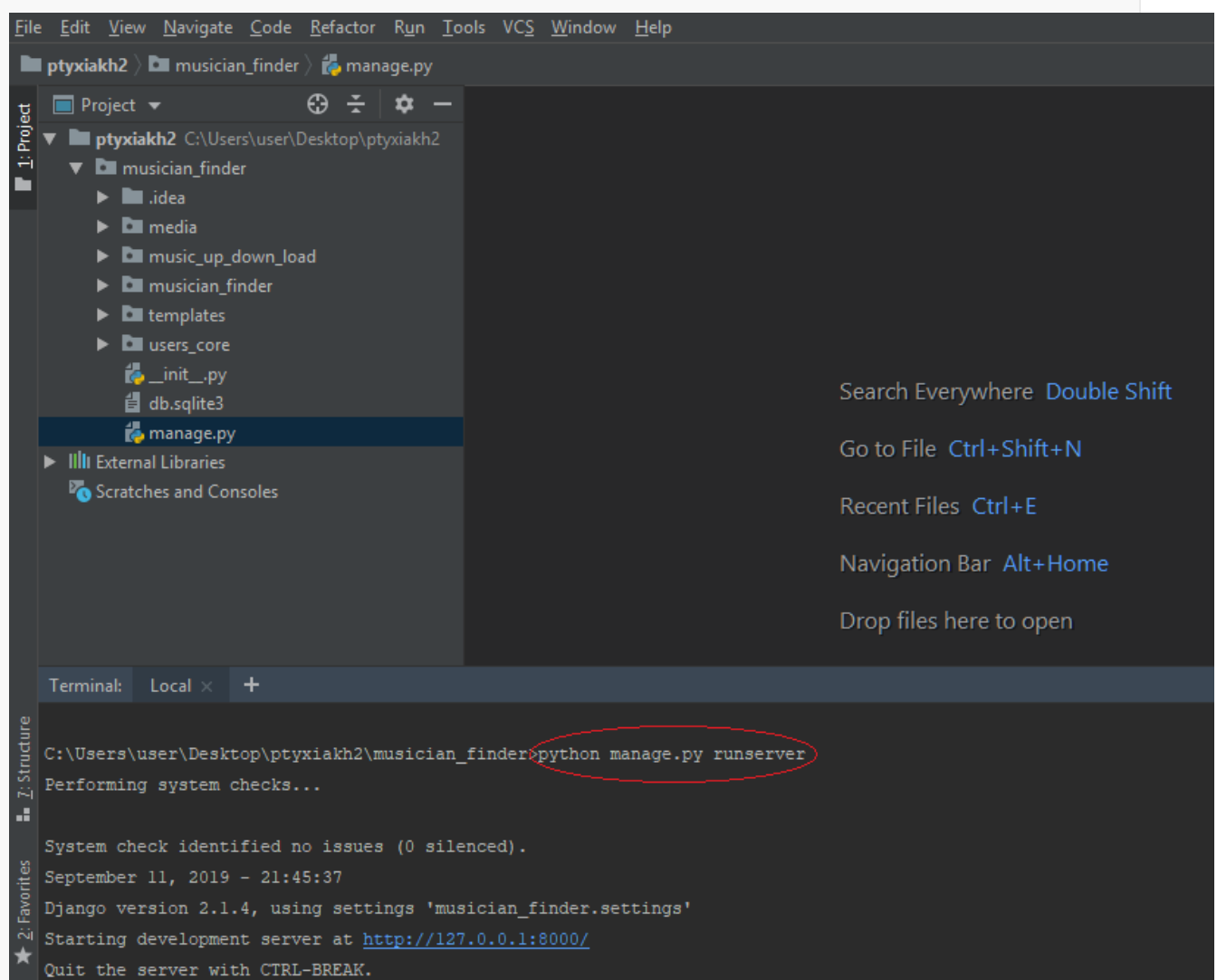
1. Αρχικά μεταφέρεται στην **home_page** ως μη εγγεγραμμένος χρήστης
2. Από την **home_page** μεταφέρεται στην **signup_page** όπου συμπληρώνει τα απαραίτητα στοιχεία για την εγγραφή του
3. Από την **signup_page** μεταφέρεται στην πάλι στην **home_page** ως εγγεγραμμένος χρήστης
4. Από την **home_page** μεταφέρεται στην **profile_page** όπου βλέπει τα στοιχεία που αφορούν το προφίλ και κάνει αλλαγές και προσθήκες σε αυτά
5. Από την **profile_page** μεταφέρεται στην **song_list_page** όπου βλέπει την λίστα με τις συνθέσεις του, η οποία είναι άδεια, και προσθέτει, διαγράφει, κατεβάζει και δημοσιοποιεί αχρεία συνθέσεων
6. Από την **song_list_page** μεταφέρεται στην **lyrics_list_page** όπου ακολουθείται η ίδια διαδικασία με την **song_list_page** αλλά με αρχεία στίχων.
7. Από την **song_list_page** μεταφέρεται στην **profile_list_page** μέσω της οποίας αναζητά άλλους μουσικούς επιλέγοντας τα μουσικά ακούσματα ή τις μουσικές ιδιότητες που θέλει να έχουν οι μουσικοί που θα βρει. Επίσης, δοκιμάζει να αναζητήσει συγκεκριμένο μουσικό συμπληρώνοντας ένα πεδίο που αφορά το όνομα χρήστη ενός μουσικού.
8. Από την **profile_list_page** επιλέγει να επισκεφθεί κάποιο προφίλ και μεταφέρεται στην **profile_appearance_page** όπου βλέπει τα στοιχεία του προφίλ του μουσικού που επέλεξε
9. Από την **profile_appearance_page** μεταφέρεται στην **other_song_list_page** όπου βλέπει την λίστα με τα αρχεία συνθέσεων του προφίλ του μουσικού που επέλεξε
10. Από την **other_song_list_page** επιστρέφει στην **profile_appearance_page** και μεταφέρεται στην **other_lyrics_list_page** όπου βλέπει την λίστα με τα αρχεία στίχων του προφίλ του μουσικού που επέλεξε
11. Από την **other_lyrics_list_page** επιστρέφει στην **profile_appearance** και μεταφέρεται στην **message_page** όπου στέλνει μήνυμα στο profile που επισκέφθηκε.
12. Μετά η παρουσίαση γίνεται από τα μάτια του χρήστη που δέχθηκε το παραπάνω μήνυμα, ανοίγοντάς το από το **inbox** και στέλνοντας απάντηση.
13. Μετά βλέπουμε πάλι τον αρχικό χρήστη να βλέπει τη συνομιλία με τους και ύστερα να κάνει logout και μετά login στην **login_page**

4.3.1 Εκκίνηση του server

Για να εκκινήσουμε τον server ανοίγουμε το PyCharm, το οποίο έχει την επιλογή terminal που μπορούμε να εκτελούμε εντολές στην command line , βρίσκουμε στο terminal τον φάκελο του project μας, δηλαδή τον φάκελο που περιέχει το αρχείο manage.py(στην περίπτωση μας ο φάκελος φέρει όνομα musician_finder), και τρέχουμε την εντολή

```
python manage.py runserver
```

Γίνονται κάποια system checks και εφόσον όλα λειτουργούν σωστά μπορούμε να επισκεφθούμε την ιστοσελίδα μας από τον browser μας στην διεύθυνση <http://127.0.0.1:8000/>.



Απόσπασμα Παρουσίασης Εφαρμογής 1: runserver

4.3.2 Home_page πριν την εγγραφή

Μόλις ζητήσουμε το url <http://127.0.0.1:8000/> , στο terminal μπορούμε να δούμε το request του browser μας στον server για το συγκεκριμένο url:

```
Terminal: Local x +
Performing system checks...

System check identified no issues (0 silenced).
September 12, 2019 - 00:12:06
Django version 2.1.4, using settings 'musician_finder.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[12/Sep/2019 00:12:17] "GET / HTTP/1.1" 200 8907
```

Απόσπασμα Παρουσίασης Εφαρμογής 2: home_page_request

Σε κάθε περίπτωση που στο terminal εμφανίζεται μόνο το request για κάποιο url χωρίς να ακολουθείται από Not found “ κάποιιο url ” τότε σημαίνει πώς υπήρξε html response για το συγκεκριμένο request.

Ο τρόπος για να καθοριστεί κάθε φορά που ζητείται το παραπάνω url να δίνεται ως response η home.html είναι το εξής urlpattern (python module που αποτελείται από python κώδικα που συσχετίζει ένα url με ένα view(βλ 3.2.4)) στο **musician_finder.urls.py**:

```
path('', TemplateView.as_view(template_name='home.html'), name='home'),
```

Στο παραπάνω urlpattern βλέπουμε από τις εξής παραμέτρους από αριστερά προς τα δεξιά:

1. Το url ''
2. Την κλάση TemplateView.as_view που λέει στο Django να αντιμετωπίζει το template home.html σαν view
3. Το namespace του urlpattern που χρησιμοποιείται σαν αντιπροσωπευτικό αυτού για λόγους ευχρηστίας.

Στον browser βλέπουμε την **home_page** της εφαρμογής μας με τον τρόπο που εμφανίζεται στην περίπτωση που την επισκεφθεί ένας μη εγγεγραμμένος χρήστης:



Απόσπασμα Παρουσίασης Εφαρμογής 3: home_page

Ο τρόπος που εμφανίζεται η **home_page** στον χρήστη καθορίζεται στο αρχείο **home.html** εισάγοντας την if statement `{% if user.is_authenticated %}` -η οποία εξετάζει αν ο χρήστης που καλεί το συγκεκριμένο html αρχείο είναι αυθεντικοποιημένος- ως εξής

```
{% if user.is_authenticated %}

    <div style="height:630px;margin-top:0px; background-size: 100%;
    background-image: url({% static 'images/musicians1.jpg' %});" >
        <div align="center" style="padding-top:200px"><h1 style="color:
#0892D0; font-style: oblique; font-weight: 900;">Εδώ θα βρείς
        τους μουσικούς που ψάχνεις!!</h1></div>

        <div align="center" style="padding-top:30px">
            <a href="{% url 'profile_search' %}" class="myButton" >Βρες
μουσικούς</a>
        </div>

    </div>

{% else %}
<div align="center" style="background-image: url({% static
'images/musicians7.jpg' %}); background-size:100%; height:630px">
    <div style="">
        <p style="padding-top:50px; font-size:20px; color:white;">Καλώς ήρθατε
στο <br>
        <h1 style="color:red; font-family:Georgia, serif">Musician
Finder!</h1><br><p style="font-size:20px; color:white;">Εδώ μπορείτε να
βρείτε μουσικούς,<br>
        να σας βρουν μουσικοί<br> και να ανταλλάξετε μηνύματα, συνθέσεις
και στίχους <br>
        <span>Αν έχετε λογαριασμό <a href="{% url 'login'
%}">Σύνδεθείτε</a> αλλιώς κάντε
        <a href="{% url 'users_core:signup' %}">Εγγραφή</a> </span></p>
    </div>
</div>
```

```

</div>
<script>
    document.getElementById("navbar").style.visibility = "hidden";
</script>

{% endif %}

```

Απόσπασμα Παρουσίασης Εφαρμογής 4: home_page if user.is_authenticated

Να σημειωθεί ότι έχει καθοριστεί μέσω του **settings.py** ο φάκελος 'templates' από τον οποίο τα views βρίσκουν τα templates:

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'postman.context_processors.inbox',
            ],
        },
    ],
]

```

Απόσπασμα Παρουσίασης Εφαρμογής 5: templates fol

4.3.3 Sign_up page

Από την **home_page** επιλέγουμε το link «Εγγραφή» το οποίο μας μεταφέρει στην **sign_up_page** και βλέπει την εξής φόρμα:

Κάνε εγγραφή στο
Musician Finder!

Όνομα Χρήστη

Απαιτείται: 150 χαρακτήρες ή λιγότεροι

Όνομα

Προαιρετικά

Επίθετο

Προαιρετικά

Email

Απαιτείται. Εισάγετε μια έγκυρη διεύθυνση email(πχ. someone@mail.com)

Κωδικός Πρόσβασης

Επιβεβαίωση Κωδικού Πρόσβασης

Εγγραφή

Απόσπασμα Παρουσίασης Εφαρμογής 5: signup_page

Για να γίνει αυτή η μεταφορά χρησιμοποιούνται δύο urlpatterns.

Το πρώτο “γεφυρώνει” το url “user/” με την εφαρμογή μας **users_core**. Αυτό σημαίνει πως ό,τι url εισάγει ο χρήστης μετά το ‘user’ καλεί views της εφαρμογής **users_core**

```
path('user/', include('users_core.urls')),
```

Το δεύτερο καλεί από το **users_core.views.py** την view συνάρτηση **signup**

```
path('signup/', views.signup, name='signup'),
```

Στην φόρμα στην **signup_page** ο χρήστης συμπληρώνει όλα τα πεδία. Τα πεδία που χρειάζονται για την δημιουργία ενός χρήστη σύμφωνα με το default user model του Django είναι υποχρεωτικά και είναι τα εξής: Όνομα χρήστη, Κωδικός ,Επαλήθευση κωδικού. Αν δεν συμπληρωθεί κάποιο πεδίο από αυτά ή συμπληρωθεί κάποιο πεδίο λάθος(πχ email address χωρίς να ακολουθεί το μοτίβο someone@address) και ο χρήστης πατήσει στο Link «Εγγραφή» που βρίσκεται στο κάτω μέρος της φόρμας τότε εμφανίζεται error που απαιτεί την συμπλήρωση αυτού του πεδίου.

Αν τα πεδία συμπληρωθούν σωστά τότε δημιουργείται ένα νέο object User σύμφωνα με τα στοιχεία που εισήγαγε στη φόρμα και αποθηκεύεται με τη συνάρτηση **save()** στη βάση μας. Η παραπάνω φόρμα έχει οριστεί ως **SignUpForm** στο αρχείο **forms.py** και αποτελεί επέκταση της φόρμας **UserCreationForm** που χρησιμοποιείται 'ώστε τα πεδία του default User model να συμπληρώνονται απευθείας μέσω αυτής. Για να εισαχθεί η φόρμα στο Html αρχείο που επιστρέφεται ως response χρησιμοποιείται η συνάρτηση

```
return render(request, 'profile.html', context)
```

Μετά επαληθεύεται εάν το object User που δημιουργήθηκε αντιστοιχεί σε object User στη βάση μας και εφόσον η επαλήθευση είναι επιτυχής τότε ο χρήστης επισυνάπτεται στη τρέχουσα σύνοδο με την συνάρτηση **login()**. Τέλος, γίνεται redirect στην **home_page** η οποία έχει οριστεί στο **settings.py** ως τη σελίδα που θα μεταφέρεται ένας χρήστης όταν κάνει login. Παρακάτω βλέπουμε τον κώδικα της view συνάρτησης **signup**, την **SignUpForm** και την παραπάνω ρύθμιση στο **settings.py**:

```
def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)

        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('home')
        else:
            form = SignUpForm()
            return render(request, 'signup.html', {'form': form})
```

Απόσπασμα Παρουσίασης Εφαρμογής 6: signup_view

```
LOGIN_REDIRECT_URL = 'home'
```

Απόσπασμα Παρουσίασης Εφαρμογής 7: login_redirect

```
class SignUpForm(UserCreationForm):
    first_name = forms.CharField(max_length=30, required=False,
    help_text='Optional.', label='Όνομα')
    last_name = forms.CharField(max_length=30, required=False,
    help_text='Optional.', label='Επίθετο')
    email = forms.EmailField(max_length=254, help_text='Required. Inform a
    valid email address.', label='Email')

    password1 = forms.CharField(label='Κωδικός Πρόσβασης',
    widget=forms.PasswordInput)
    password2 = forms.CharField(label='Επιβεβαίωση Κωδικού Πρόσβασης',
    widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email',
        'password1', 'password2')
        labels = {'username': "Όνομα Χρήστη", }
```

Απόσπασμα Παρουσίασης Εφαρμογής 8: signup_form

Μόλις δημιουργηθεί ένα user object αυτομάτως δημιουργείται και ένα Profile object που σχετίζεται με αυτό. Αυτό γίνεται χρησιμοποιώντας τον “singal dispatcher” του Django. Συγκεκριμένα, όταν δημιουργείται ένας user τότε δημιουργείται profile για τον συγκεκριμένο user και μετά αποθηκεύεται στη βάση. Ο κώδικας είναι ο εξής:

```
@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()
```

Απόσπασμα Παρουσίασης Εφαρμογής 9: singals

Να σημειωθεί ότι για να λειτουργήσει το παραπάνω πρέπει να εισάγουμε στο **apps.py** την κλάση:

```
class UsersCoreConfig(AppConfig):
    name = 'users_core'

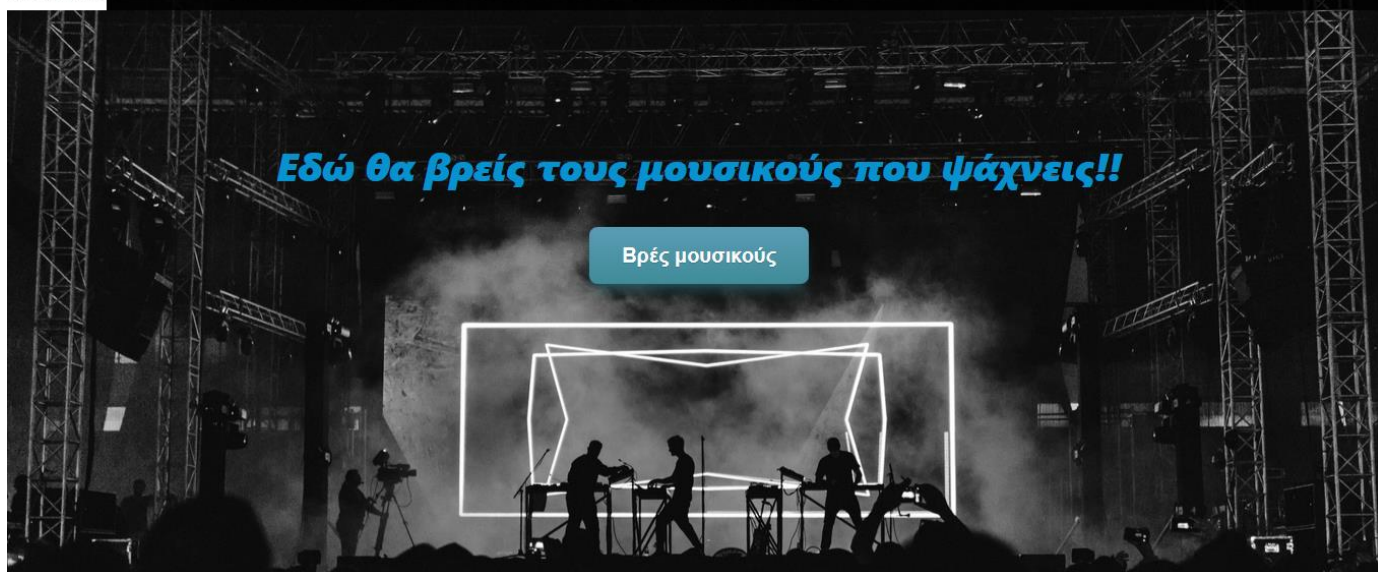
    def ready(self):
        from . import singals
```

Απόσπασμα Παρουσίασης Εφαρμογής 10: UsersCoreConfig

Κάθε user object αντιστοιχεί σε ένα profile object και αντίστροφα. Αν διαγραφεί ένα user object τότε διαγράφεται και το αντίστοιχο profile object. Το αντίστροφο όμως δεν ισχύει.

4.3.4 Home page μετά την εγγραφή

Ο χρήστης ως αυθεντικοποιημένος πλέον στην **home_page** βλέπει το εξής:



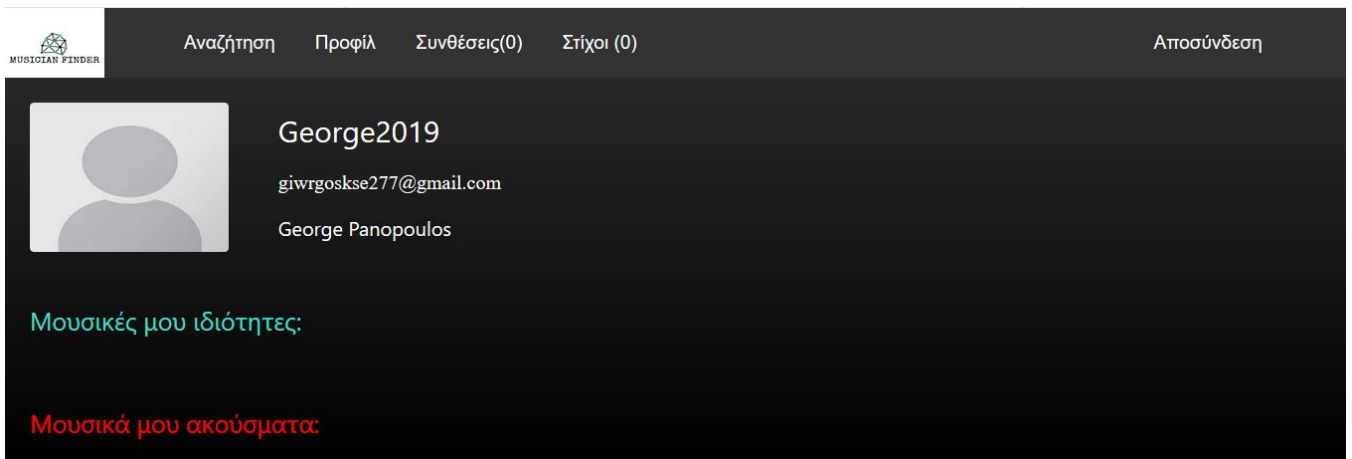
Απόσπασμα Παρουσίασης Εφαρμογής 12: home_page2

Στην παραπάνω εικόνα φαίνονται τα Links που μπορεί να επισκεφθεί. Το κάθε ένα από αυτά έχει τον εξής σκοπό:

1. «Αναζήτηση»: Οδηγεί στην σελίδα όπου μπορεί να αναζητήσει άλλους μουσικούς
2. «Πρόφιλ»: Οδηγεί στην σελίδα του προφίλ του που μπορεί να κάνει αλλαγές και προσθήκες σε αυτήν
3. «Συνθέσεις»: Οδηγεί στην σελίδα που μπορεί να ανεβάσει να διαγράψει και να δημοσιεύσει αχρεία συνθέσεων και να δει τις συνθέσεις που έχει ήδη ανεβάσει
4. «Στίχοι»: Οδηγεί στην σελίδα που μπορεί να ανεβάσει να διαγράψει και να δημοσιεύσει αρχεία στίχων και να δει τους στίχους που έχει ήδη ανεβάσει
5. «Αποσύνδεση»: Αποσυνδέει τον χρήστη και οδηγεί στην σελίδα **login_page**
6. «Βρες Μουσικούς»: Οδηγεί στην σελίδα όπου μπορεί να αναζητήσει άλλους μουσικούς

4.3.5 Profile page

Από τα links του navigation bar στο πάνω μέρος της **home page** ο χρήστης επιλέγει το «Πρόφιλ» και οδηγείται στην **profile page**.



Απόσπασμα Παρουσίασης Εφαρμογής 13: profile_page_!

Το urlpattern για το παραπάνω response είναι το εξής:

```
path('profile/', user_view.profile, name='profile'),
```

Η **users_view** αναφέρεται στην εφαρμογή μας **users_core** την οποία έχουμε κάνει import στο **urls.py** του **musician_finder**:

```
from users_core import views as user_view
```

Αρχικά βλέπει την default εικόνα προφίλ του. Η εικόνα αυτή έχει καθοριστεί στο πεδίο **image** του profile model ως εξής:

```
image = models.ImageField(default='default.jpg', upload_to='profile pics')
```

Απόσπασμα Παρουσίασης Εφαρμογής 14: default_image

Η παράμετρος **default** όταν δημιουργείται ένα προφίλ ψάχνει να βρει στον φάκελο **media** εικόνα με όνομα 'default.jpg'. Η λειτουργία αυτή, όπως και το που θα αποθηκεύονται αρχεία που ανεβάζει ο χρήστης, έχει καθοριστεί στο **settings.py**:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Απόσπασμα Παρουσίασης Εφαρμογής 15: MEDIA

Στα δεξιά της εικόνας προφίλ βλέπει το username, το email, και το ονοματεπώνυμο του. Κάτω από αυτήν βλέπει τις μουσικές του ιδιότητες και τα μουσικά του ακούσματα τα οποία είναι άδεια προς το παρόν. Κάνοντας scroll down βλέπει τα πεδία με τα προσωπικά του στοιχεία που είναι ουσιαστικά αυτά που εισήγαγε στην **sign_up_page** (βλ 4.3.3) συν το πεδίο που αφορά την αλλαγή της εικόνας προφίλ. Τα πεδία αυτά μπορεί να τα επεξεργαστεί αλλάζοντάς τα και πατώντας το Link «Ανανέωση»:



Αναζήτηση Προφίλ Συνθέσεις(0) Στίχοι (0) Αποσύνδεση

Επεξεργασία στοιχείων προφίλ

Προσωπικά στοιχεία

Όνομα Χρήστη

Απαιτείται 150 χαρακτήρες ή λιγότεροι

Όνομα

Προαιρετικά

Επίθετο

Προαιρετικά

Email

Απαιτείται. Εισάγετε μια έγκυρη διεύθυνση email (ex: someone@mail.com)

Ημερομηνία Γέννησης

January

1

1940

Εικόνα Προφίλ

Currently: [default.jpg](#)

Change: No file selected.

Ανανέωση

Απόσπασμα Παρουσίασης Εφαρμογής 16: profile_1

Παρακάτω βλέπει τα Μουσικά του στοιχεία τα οποία είναι οι μουσικές ιδιότητες ου και τα μουσικά ακούσματά του. Στα στοιχεία αυτά μπορεί να κάνει αλλαγές προσθέτοντας ή αφαιρώντας μουσικές ιδιότητες ή μουσικά ακούσματα και πατώντας το link «Ανανέωση»:

Μουσικά Στοιχεία

Μουσικές Ιδιότητες

☐ Drummer
 ☐ Τραγουδιστής
 ☐ Κιθαρίστας
 ☐ Μπασίστας
 ☐ Στιχουργός
 ☐ Σαξοφωνίστας
 ☐ Κλαριτζής
 ☐ Dj

Μουσικά Ακούσματα

Rock

☐ Classic Rock
 ☐ Progressive Rock
 ☐ Post Rock
 ☐ Rock&Roll
 ☐ Psychedelic Rock
 ☐ Hard Rock
 ☐ Garage Rock
 ☐ Surf Rock
 ☐ Rockabilly

Metal

☐ Black Metal
 ☐ Death Metal
 ☐ Thrash Metal
 ☐ Doom Metal
 ☐ Progressive Metal
 ☐ Sludge Metal
 ☐ Heavy Metal
 ☐ Metalcore
 ☐ Grindcore

Punk

☐ Punk Rock
 ☐ Pop Punk
 ☐ Garage Punk
 ☐ Hardcore Punk
 ☐ Thrash Punk
 ☐ Crust Punk
 ☐ Folk Punk
 ☐ Ska Punk

Pop

☐ Indie Pop
 ☐ Synth Pop
 ☐ Power Pop
 ☐ Dream Pop
 ☐ Noise Pop
 ☐ Experimental Pop
 ☐ Jangle Pop

Blues

☐ Blues
 ☐ Blues Rock
 ☐ Country Blues
 ☐ Rythm & Blues
 ☐ Electric Blues
 ☐ Boogie-Woogie
 ☐ Bluegrass
 ☐ Delta Blues

Jazz

☐ Jazz
 ☐ Nu Jazz
 ☐ Modern Jazz
 ☐ Free Jazz
 ☐ Soul Jazz
 ☐ Swing
 ☐ Latin Jazz
 ☐ Vocal Jazz
 ☐ Spiritual Jazz

Funk

☐ Funk
 ☐ Funk Jam
 ☐ Deep Funk
 ☐ Funk Rock
 ☐ Jazz Funk
 ☐ g-Funk
 ☐ Electro Funk
 ☐ Go-go

Soul

☐ Soul
 ☐ Neo-Soul
 ☐ Gospel

Electronic

☐ House
 ☐ Electronica
 ☐ Downtempo
 ☐ Techno
 ☐ Electro
 ☐ Dubstep
 ☐ Drum & Bass
 ☐ Trance
 ☐ Dub
 ☐ Vaporwave

Hiphop rap

☐ Rap
 ☐ Trap
 ☐ Instrumental Hip-Hop
 ☐ Boom-Bap
 ☐ Beat-tape

Greek

☐ Έντεχνο
 ☐ Λαϊκό
 ☐ Ρεμπέτεκο

Other

☐ Ska
 ☐ No Wave
 ☐ Folk
 ☐ New wave
 ☐ Swing
 ☐ Gospel

[Ανανέωση](#)

Απόσπασμα Παρουσίασης Εφαρμογής 17: profile_2

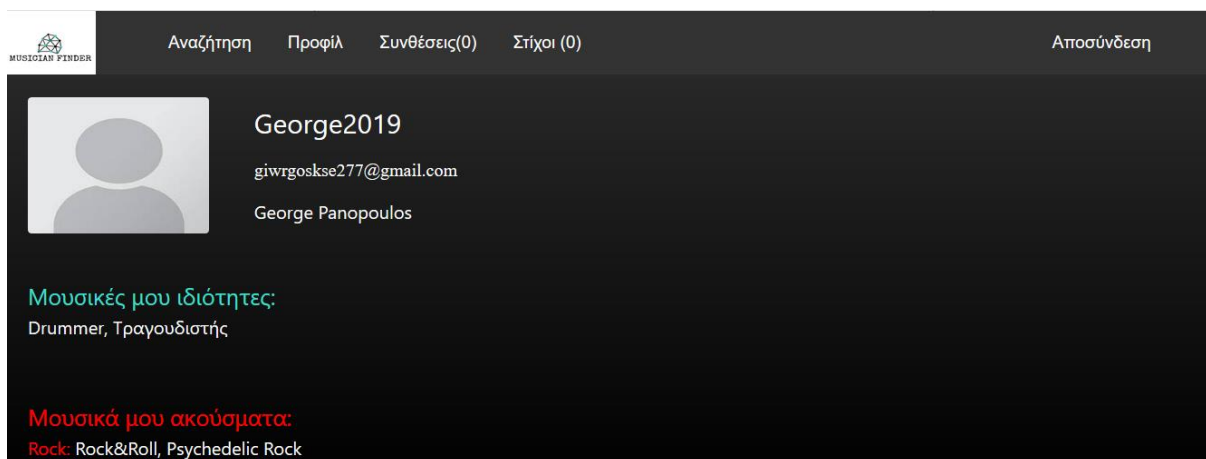
Στο παράδειγμά μας ο χρήστης επιλέγει ως μουσικές ιδιότητες τις Drummer και Τραγουδιστής και ως μουσικά ακούσματα τα Rock&Roll και Psychedelic Rock:



The screenshot shows a form with three sections: 'Μουσικές Ιδιότητες' (Musical Characteristics) with checkboxes for Drummer, Τραγουδιστής, Κιθαρίστας, Μπασιστάς, Στιχουργός, Σαξοφωνίστας, Κλαριτζής, and Dj; 'Μουσικά Ακούσματα' (Musical Sounds); and 'Rock' with checkboxes for Classic Rock, Progressive Rock, Post Rock, Rock&Roll, Psychedelic Rock, Hard Rock, Garage Rock, Surf Rock, and Rockabilly. The selected options are Drummer, Τραγουδιστής, Rock&Roll, and Psychedelic Rock.

Απόσπασμα Παρουσίασης Εφαρμογής 17: profile_3

Μετά πατά το Link «Ανανέωση» και μεταφέρεται στο αρχικό σημείο της σελίδας του προφίλ του βλέποντας πλέον τα μουσικά στοιχεία που έχει εισάγει:



The screenshot shows a user profile for 'George2019' (George Panopoulos). It displays the user's email and a section for 'Μουσικές μου ιδιότητες:' (My musical characteristics:) listing Drummer and Τραγουδιστής. Another section for 'Μουσικά μου ακούσματα:' (My musical sounds:) lists Rock, Rock&Roll, and Psychedelic Rock.

Απόσπασμα Παρουσίασης Εφαρμογής 18: profile_4

Η συνάρτηση view που παρέχει την παραπάνω λειτουργικότητα είναι η **profile**. Αυτό που κάνει είναι να εισάγει στο αρχείο **profile.html** τις φόρμες **u_form** και **p_form** που είναι για τα στοιχεία **όνομα**, **επίθετο**, **email** και τα στοιχεία **ημερομηνία γέννησης**, **εικόνα προφίλ**, **μουσικές ιδιότητες**, **μουσικά ακούσματα** αντίστοιχα. Αν δεν υπάρχουν αλλαγές εμφανίζει τα στοιχεία ως έχουν. Αν υπάρχουν αλλαγές τότε ελέγχει την εγκυρότητα των στοιχείων που εισήχθησαν σε αυτές τα αποθηκεύει στη βάση και κάνει redirect στην **profile_page**:

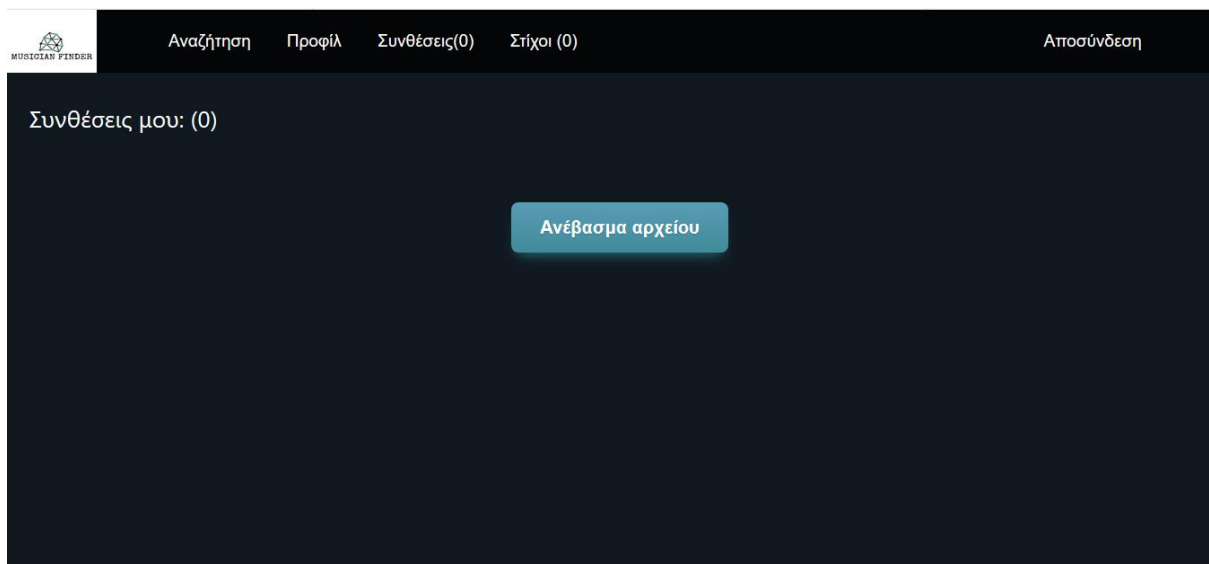
```
@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST, request.FILES,
instance=request.user.profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            return redirect('profile')
    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)
        u = UserUpdateForm
        p = ProfileUpdateForm
```

```
context = {
    'u_form': u_form,
    'p_form': p_form,
}
return render(request, 'profile.html', context)
```

Απόσπασμα Παρουσίασης Εφαρμογής 19: profile_view

4.3.6 Song list page

Από τα links του navigation bar ο χρήστης επιλέγει το «Συνθέσεις» και μεταφέρεται στην **song_list_page**.



Απόσπασμα Παρουσίασης Εφαρμογής 20: song_list_page_1

Το urlpattern που οδηγεί σε αυτή τη σελίδα είναι:

```
path('song_list/', user_view.composition_list, name='song_list'),
```

Επειδή ο χρήστης είναι καινούργιος δεν υπάρχει ακόμα κάποια ανεβασμένη σύνθεση. Επιλέγει το link «Ανέβασμα αρχείου» το οποίο τον οδηγεί στην **upload_song_page** με το urlpattern

```
path('song_list/upload', user_view.upload_song, name='upload_song'),
```

. Στην σελίδα αυτή μπορεί να ανεβάσει μια σύνθεση ,να της δώσει τίτλο καθώς και να επιλέξει εάν θα φαίνεται ή όχι στους άλλους χρήστες που θα επισκέπτονται το προφίλ του. Τα πεδία «composition title» και «composition» που αφορούν τον τίτλο της σύνθεσης και το αρχείο της σύνθεσης αντίστοιχα και είναι υποχρεωτικά. Ο τύπος αρχείου που είναι δεκτός πρέπει να είναι

ανάμεσα στα mp3, wav, flac. Σε περίπτωση που δεν είναι, εμφανίζεται μήνυμα λάθους που εξηγεί ότι το αρχείο πρέπει να ανήκει στις παραπάνω κατηγορίες. Οι κανόνες για την απαίτηση συμπλήρωσης των πεδίων έχουν καθοριστεί στην **SongForm** ενώ οι έγκυροι τύποι αρχείων στο πεδίο **composition** του μοντέλου **song** με την παράμετρο **validators** . Η επιλογή `is public` καθορίζει εάν η σύνθεση θα φαίνεται και στους άλλους χρήστες.



Απόσπασμα Παρουσίασης Εφαρμογής 20: `upload_song_page`

Ο χρήστης συμπληρώνει το πεδίο «composition title» γράφοντας «psychedelic drums». Ανεβάζει σχετικό αρχείο mp3 και επιλέγει το `is_public`. Μετά πατά το link «Ανέβασμα». Εκεί εξετάζεται η εγκυρότητα των δεδομένων που εισήχθησαν στην **SongForm**. Αν είναι έγκυρα τότε συσχετίζεται το αρχείο που ανέβηκε με τον χρήστη, αποθηκεύεται στη βάση και γίνεται `redirect` στην **song_list_page**.

Η view συνάρτηση που παρέχει την παραπάνω λειτουργικότητα είναι η `upload_song`:

```
@login_required
def upload_song(request):
    if request.method == 'POST':
        form = SongForm(request.POST, request.FILES)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.user = request.user
            instance.save()
            return redirect('song_list')
        else:
            form = SongForm()

    return render(request, 'upload_song.html', {'form': form})
```

Απόσπασμα Παρουσίασης Εφαρμογής 21: `upload_song`

Η **SongForm** είναι η εξής:

```
class SongForm(forms.ModelForm):

    def __init__(self, *args, **kwargs):
```

```

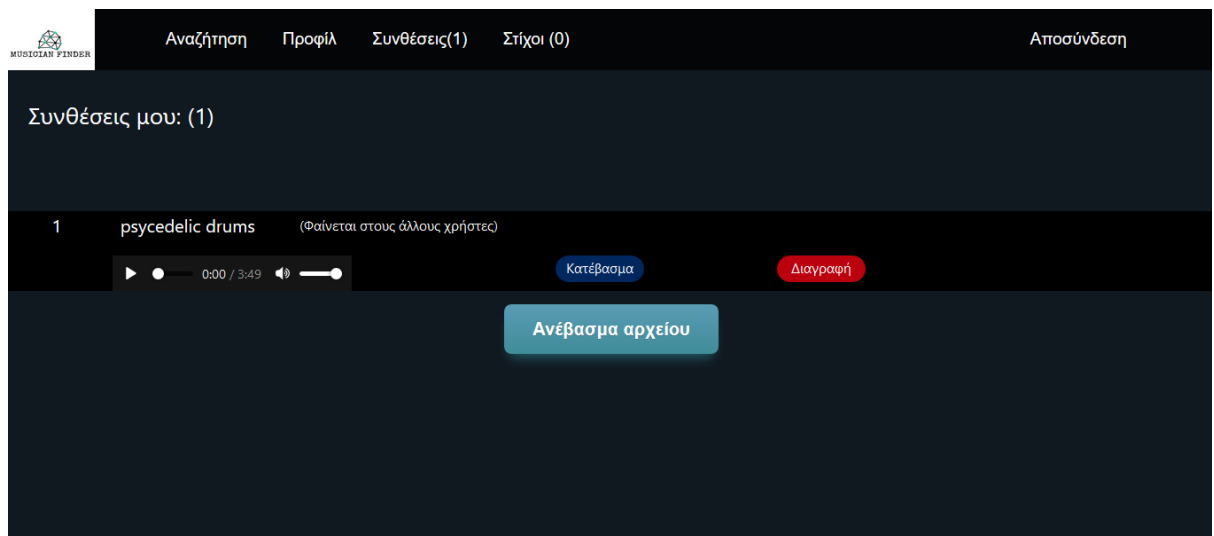
super(SongForm, self).__init__(*args, **kwargs)
self.fields['composition'].required = True
self.fields['composition_title'].required = True

class Meta:
    model = Song
    fields = ('composition_title', 'composition', 'is_public')

```

Απόσπασμα Παρουσίασης Εφαρμογής 22: SongForm

Στην **song_list_page** μπορεί να δει την σύνθεση που ανέβασε ,να την ακούσει, να την κατεβάσει ή να την διαγράψει.



Απόσπασμα Παρουσίασης Εφαρμογής 21: song_list_page_2

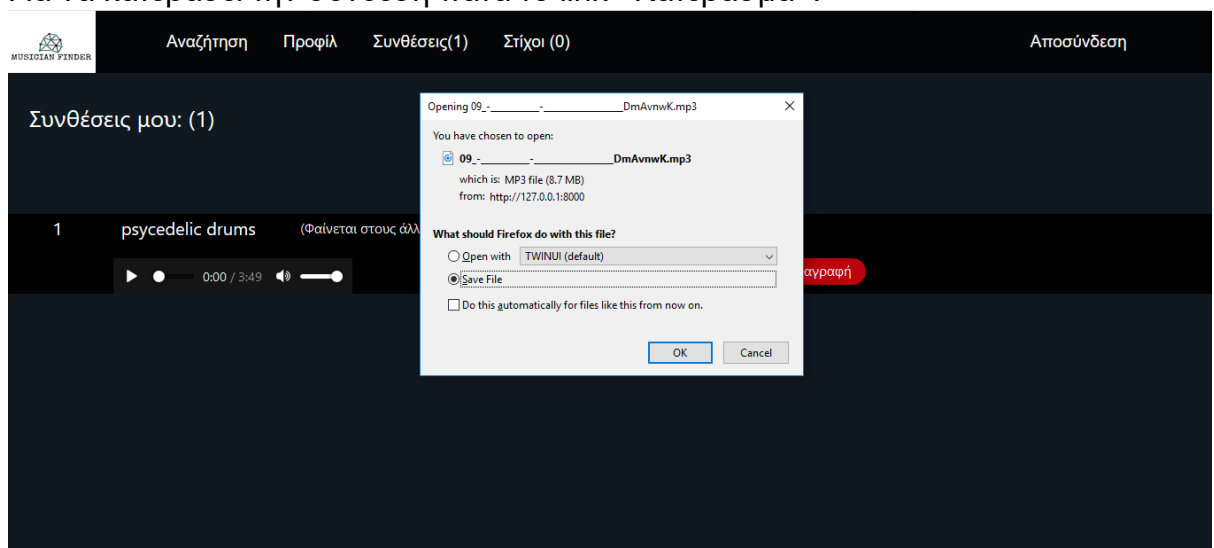
Το urlpatternr για την παραπάνω μεταφορά είναι:

```

path('song_delete/<int:song_id>', user_view.song_delete,
name='song_delete'),

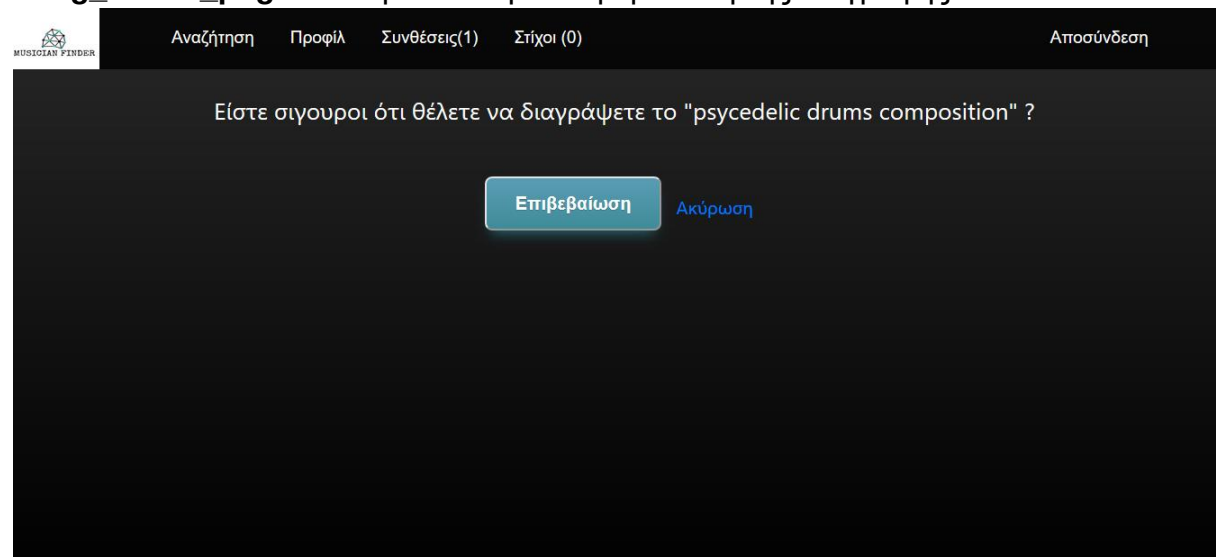
```

Για να κατεβάσει την σύνθεση πατά το link «Κατέβασμα»:



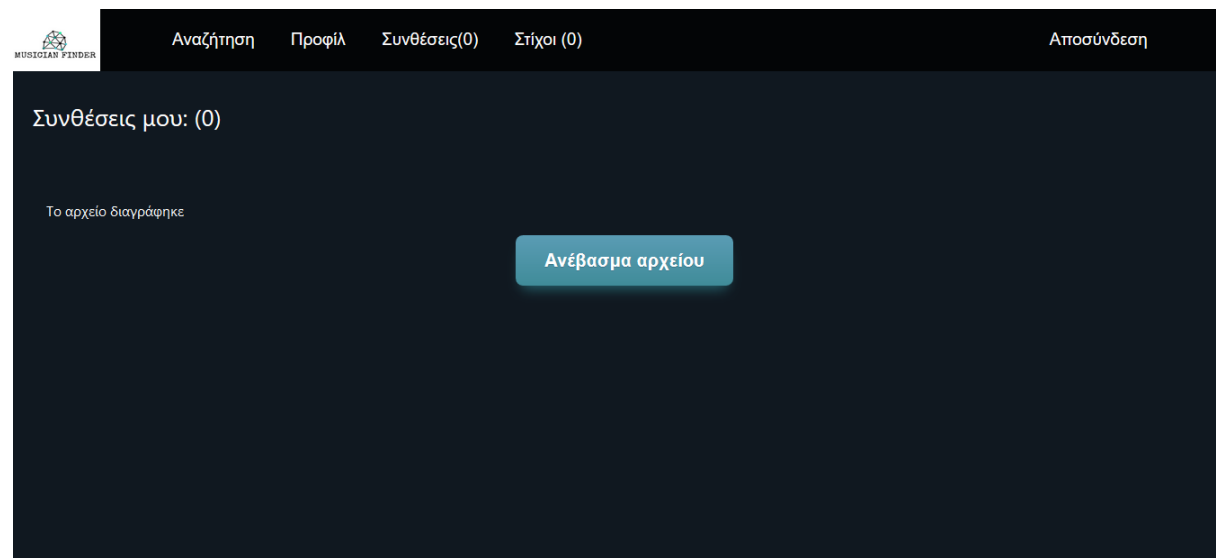
Απόσπασμα Παρουσίασης Εφαρμογής 21: song_download

Για να διαγράψει την σύνθεση πατά το link «Διαγραφή» και μεταφέρεται στην **song_delete_page** που ερωτάται για επιβεβαίωση της διαγραφής:



Απόσπασμα Παρουσίασης Εφαρμογής 22: song_confirm_deletion

Αν πατήσει «Ακύρωση» επιστρέφει στην **song_list_page** χωρίς να διαγραφεί η σύνθεση. Αν πατήσει «Επιβεβαίωση» τότε διαγράφεται η σύνθεση και γίνεται redirect στην **song_list_page** όπου εμφανίζεται μήνυμα επιβεβαίωσης της διαγραφής και η λίστα είναι πάλι άδεια:



Απόσπασμα Παρουσίασης Εφαρμογής 23: song_deletion

Η view function που παρέχει την παραπάνω λειτουργικότητα είναι η **song_delete** και έχει τον εξής κώδικα:

```
@login_required
def song_delete(request, song_id):
    song = get_object_or_404(Song, pk= song_id)
```

```

if request.method == "POST":

    song.delete()
    messages.success(request, 'Το αρχείο διαγράφηκε')
    return redirect('song_list')

return render(request, 'song_delete.html', {'song': song})

```

Απόσπασμα Παρουσίασης Εφαρμογής 24: song_delete

Η ίδια διαδικασία μπορεί να ακολουθηθεί για τα αρχεία στίχων με τη διαφορά ότι ο επιτρεπτός τύπος αρχείου είναι pdf.

4.3.7 Profile list page

Από την **song_list_page** μεταφέρεται στην **profile_list_page** μέσω του links «Αναζήτηση» στο navigation bar:

Μουσικές ιδιότητες: ☐ Drummer ☐ Τραγουδιστής ☐ Κιθαρίστας ☐ Μπασίστας ☐ Στιχουργός ☐ Σαξοφωνίστας ☐ Κλαρινετής ☐ Dj

Μουσικά ακούσματα:

Rock: ☐ Classic Rock ☐ Progressive Rock ☐ Post Rock ☐ Rock&Roll ☐ Psychedelic Rock ☐ Hard Rock ☐ Garage Rock ☐ Surf Rock ☐ Rockabilly

Metal: ☐ Black Metal ☐ Death Metal ☐ Thrash Metal ☐ Doom Metal ☐ Progressive Metal ☐ Sludge Metal ☐ Heavy Metal ☐ Metalcore ☐ Grindcore

Punk: ☐ Punk Rock ☐ Pop Punk ☐ Garage Punk ☐ Hardcore Punk ☐ Thrash Punk ☐ Crust Punk ☐ Folk Punk ☐ Ska Punk

Pop: ☐ Indie Pop ☐ Synth Pop ☐ Power Pop ☐ Dream Pop ☐ Noise Pop ☐ Experimental Pop ☐ Jangle Pop

Blues: ☐ Blues ☐ Blues Rock ☐ Country Blues ☐ Rhythm & Blues ☐ Electric Blues ☐ Boogie-Woogie ☐ Bluegrass ☐ Delta Blues

Jazz: ☐ Jazz ☐ Nu Jazz ☐ Modern Jazz ☐ Free Jazz ☐ Soul Jazz ☐ Swing ☐ Latin Jazz ☐ Vocal Jazz ☐ Spiritual Jazz

Funk: ☐ Funk ☐ Funk Jam ☐ Deep Funk ☐ Funk Rock ☐ Jazz Funk ☐ g-Funk ☐ Electro Funk ☐ Go-go

Soul: ☐ Soul ☐ Neo-Soul ☐ Gospel

Electronic: ☐ House ☐ Electronica ☐ Downtempo ☐ Techno ☐ Electro ☐ Dubstep ☐ Drum & Bass ☐ Trance ☐ Dub ☐ Vaporwave

Hip-hop rap: ☐ Rap ☐ Trap ☐ Instrumental Hip-Hop ☐ Boom-Bap ☐ Beat-tape

Greek: ☐ Έντεχνο ☐ Λαϊκό ☐ Ρεμπέτικο

Other: ☐ Ska ☐ No Wave ☐ Folk ☐ New wave ☐ Swing ☐ Gospel

Όνομα χρήστη:

Αναζήτηση

Απόσπασμα Παρουσίασης Εφαρμογής 25: profile_list_1

Το urlpattern για την παραπάνω ,μεταφορά είναι:

```
path('search/', user_view.profile_search, name='profile_search'),
```

Εδώ μπορεί να αναζητήσει μουσικούς συμπληρώνοντας τουλάχιστον ένα από τα πεδία «Μουσικές Ιδιότητες» και «Μουσικά ακούσματα». Εάν θελήσει να αναζητήσει κάποιο προφίλ μέσω του username του συμπληρώνει το πεδίο «Όνομα χρήστη». Στο παράδειγμά μας ο χρήστης επιλέγει να αναζητήσει μουσικό με μουσικές ιδιότητες τραγουδιστής, κιθαρίστας και μουσικά ακούσματα classic rock και psychedelic rock:

Απόσπασμα Παρουσίασης Εφαρμογής 26: profile_list_choices

Μετά πατά το Link «Αναζήτηση» που βρίσκεται στο κάτω μέρος της σελίδας και εμφανίζεται -στην ίδια σελίδα- προφίλ σύμφωνα με τα μουσικά στοιχεία που επέλεξε:

Απόσπασμα Παρουσίασης Εφαρμογής 27: profile_list_results

Αυτό που έγινε είναι ότι ο χρήστης ζήτησε ανάμεσα σε όλα τα profile objects να εμφανιστούν αυτά που έχουν τις παραμέτρους που εισήγαγε

Για να γίνει αυτό χρησιμοποιήθηκε η εφαρμογή του Django **django-filters** την οποία έκανα Import στο αρχείο **filters.py** όπου έβαλα φίλτρο για κάθε πεδίο του profile μοντέλου που σχετίζεται με τις μουσικές ιδιότητες και τα μουσικά είδη το οποίο καλεί συνάρτηση που χρησιμοποιείται για να «φιλτράρει» το κάθε πεδία ανάλογα με την τιμή που εισάγει ο χρήστης. Παρακάτω φαίνεται το φίλτρο για το πεδίο **rock** και η συνάρτηση που καλείται από το συγκεκριμένο φίλτρο.

```
class ProfileFilter(django_filters.FilterSet):  
  
    rock =  
    django_filters.MultipleChoiceFilter(choices=Profile.ROCK_CHOICES,  
    required=False,
```

```

widget=forms.CheckboxSelectMultiple(),
method='rock_filter'
)

```

Απόσπασμα Παρουσίασης Εφαρμογής 28: profile_filter

Το φίλτρο που εισάχθηκε είναι **MultipleChoiceFilter** διότι θέλουμε να φιλτράρουμε πεδίο **MultipleSelectField**. Κάθε τύπος πεδίου σχετίζεται με διαφορετικό φίλτρο

```

def rock_filter(self, queryset, name, value):
    print(value)
    choice = value
    return queryset.filter(rock=value)

```

Απόσπασμα Παρουσίασης Εφαρμογής 29: profile_filter_2

Τα παραπάνω ισχύουν αντίστοιχα και για τα υπόλοιπα μουσικά πεδία.

Η view συνάρτηση που εισάγει τα φίλτρα στα profile objects και εμφανίζει το αποτέλεσμα του «φιλτραρίσματος» στο template μας είναι η **profile_search** και έχει τον εξής κώδικα:

```

@login_required
def profile_search(request):

    profile_list = Profile.objects.all()
    profile_filter = ProfileFilter(request.GET, queryset=profile_list)
    profile_count = ProfileFilter(request.GET,
    queryset=profile_list).qs.count()
    submit = request.GET.get('Submit', False)

    return render(request, 'profile_list.html', {'filter': profile_filter,
    'submit': submit, 'profiles': profile_count })

```

Απόσπασμα Παρουσίασης Εφαρμογής 29: profile_search

Επίσης, ο χρήστης μπορεί να εισάγει στο πεδίο «Όνομα χρήστη» κάποιο **username** ώστε να βρει κάποιον χρήστη που γνωρίζει το username του. Με αυτόν τον τρόπο φιλτράρονται όλα τα profile objects σύμφωνα με το username που σχετίζονται:

```

user_username = django_filters.CharFilter(label='Username',
lookup_expr="contains")

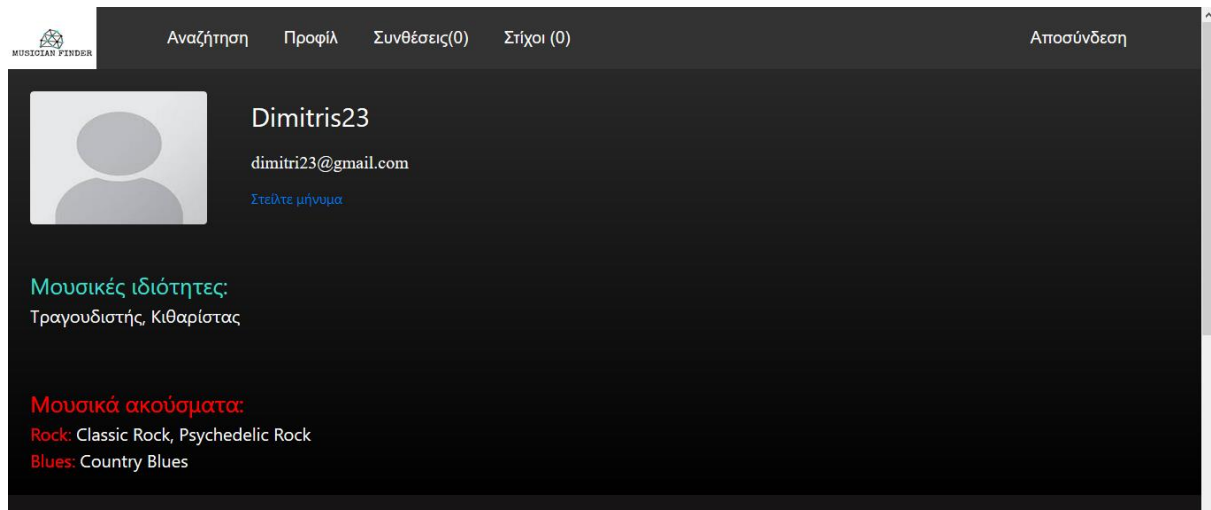
```

Απόσπασμα Παρουσίασης Εφαρμογής 30: profile_filter_username

Το φίλτρο που εισάχθηκε είναι **CharFilter** διότι θέλουμε να φιλτράρουμε πεδίο **CharField**

4.3.8 Profile appearance page

Πατώντας το link «Dimitris23» που βρίσκεται πάνω στο αποτέλεσμα για το προφίλ που αναζήτησε ο χρήστης (βλ 4.3.7), από την **profile_list_page** μεταφέρεται στην **profile_appearance_page** του προφίλ Dimitris23 που είναι ουσιαστικά το πώς φαίνεται το προφίλ του σε άλλους χρήστες:



Απόσπασμα Παρουσίασης Εφαρμογής 31: profile_appearance_1

Το urlpattern για την παραπάνω ,μεταφορά είναι:

```
path('profile/<int:prof_id>', user_view.profile_appearance,
name='profile_appearance'),
```

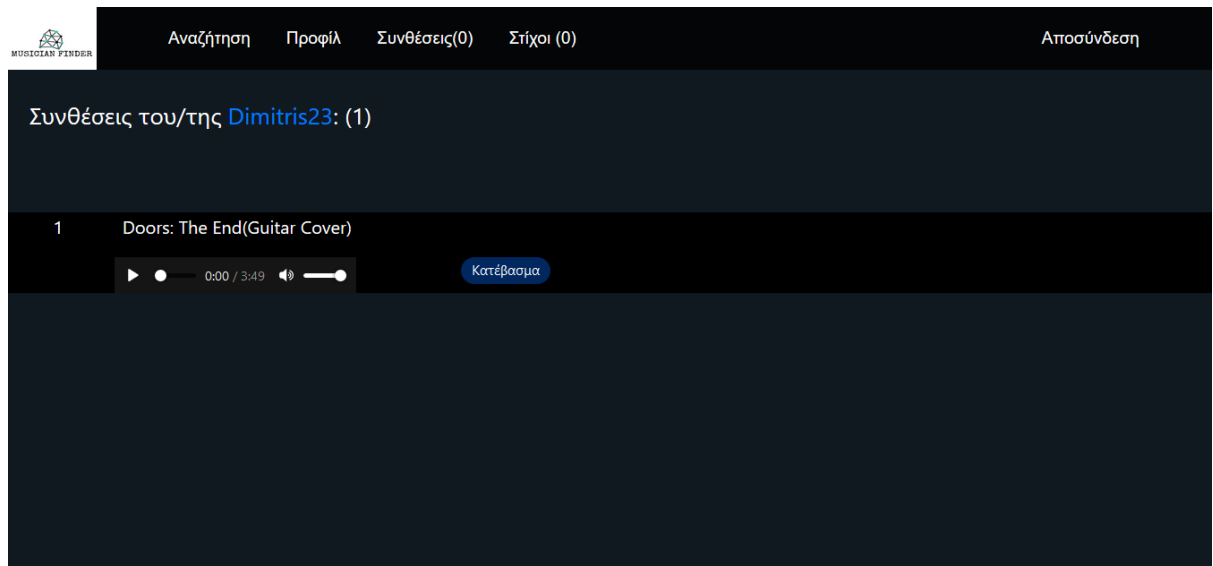
Αρχικά ο χρήστης βλέπει το username, το email ,τις μουσικές ιδιότητες και όλα τα μουσικά ακούσματα του Dimitris23. Επίσης υπάρχει «Στείλε μήνυμα » μέσω του οποίου μπορεί να επικοινωνήσει μαζί του. Κάνοντας scroll down βλέπει τα υπόλοιπα στοιχεία του προφίλ του καθώς και links που οδηγούν στις σελίδες με τους στίχους και τις συνθέσεις του:



Απόσπασμα Παρουσίασης Εφαρμογής 32: profile_appearance_2

4.3.9 Other song list page

Από την **profile_appearance_page** του προφίλ Dimitris23 ο χρήστης πατά το link «Συνθέσεις» και μεταφέρεται στην **other_song_list_page**.



Απόσπασμα Παρουσίασης Εφαρμογής 33: other_song_list_page

Το urlpattern για την παραπάνω μεταφορά είναι:

```
path('song_list/<int:prof_id>', user_view.other_composition_list,  
name='other_song_list'),
```

Εκεί ο χρήστης μπορεί να ακούσει και να κατεβάσει την σύνθεση του Dimitris23.

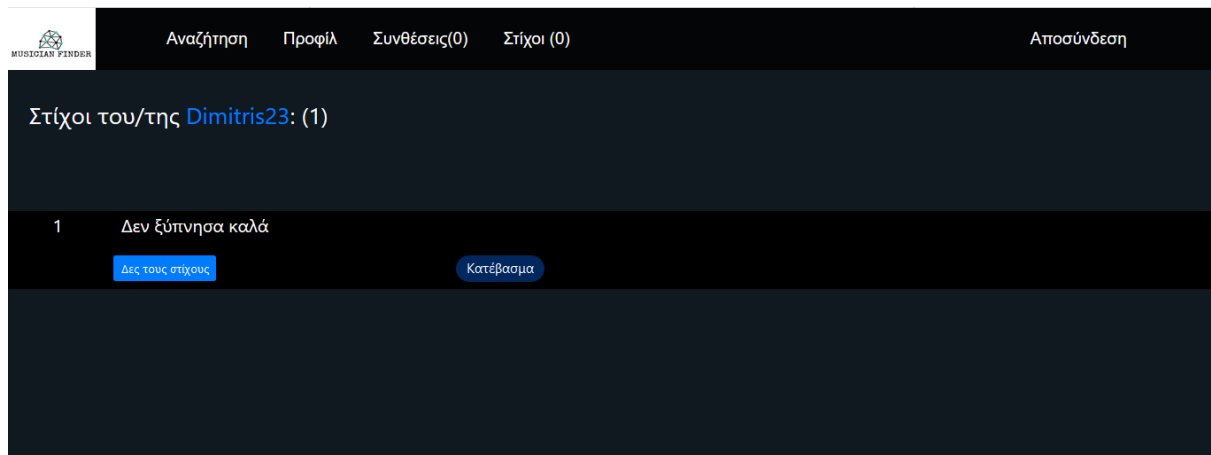
Η view συνάρτηση για την παραπάνω λίστα είναι η **other_composition_list** και έχει τον εξής κώδικα:

```
@login_required  
def other_composition_list(request, prof_id):  
    prof = get_object_or_404(Profile, pk=prof_id)  
    return render(request, 'other_song_list.html', {'profile': prof})
```

Απόσπασμα Παρουσίασης Εφαρμογής 343: other_composition_list

4.3.10 Other lyrics list page

Από την **other_song_list_page** του Dimitris23 μεταφέρεται στην **profile_appearance_page** μέσω του link «Dimitris23» και μετά επιλέγει το Link «Στίχοι» ώστε να δει την λίστα με τους στίχους του Dimitris23:



Απόσπασμα Παρουσίασης Εφαρμογής 35: other_song_list_page

Το urlpattern για την παραπάνω μεταφορά είναι:

```
path('lyrics_list/<int:prof_id>', user_view.other_lyrics_list,  
name='other_lyrics_list'),
```

Εκεί ο χρήστης μπορεί να ακούσει και να κατεβάσει το αρχείο στίχων του Dimitris23. Μέσω του link «Δες τους στίχους» μπορεί να δει το αρχείο PDF που ανέβασε ο Dimitris23 με το περιεχόμενο των στίχων:

Δεν ξύπνησα καλά

Σηκώθηκα νωρίς
Και ας ήμουν κουρασμένος
Χθες το βραδύ ξέχασα
Πως είμαι πονεμένος

Σηκώθηκα νωρίς
Και ας ήμουν κουρασμένος
Χθες το βραδύ ξέχασα
Πως είμαι πονεμένος

Απόσπασμα Παρουσίασης Εφαρμογής 36: other_song_list_PDF

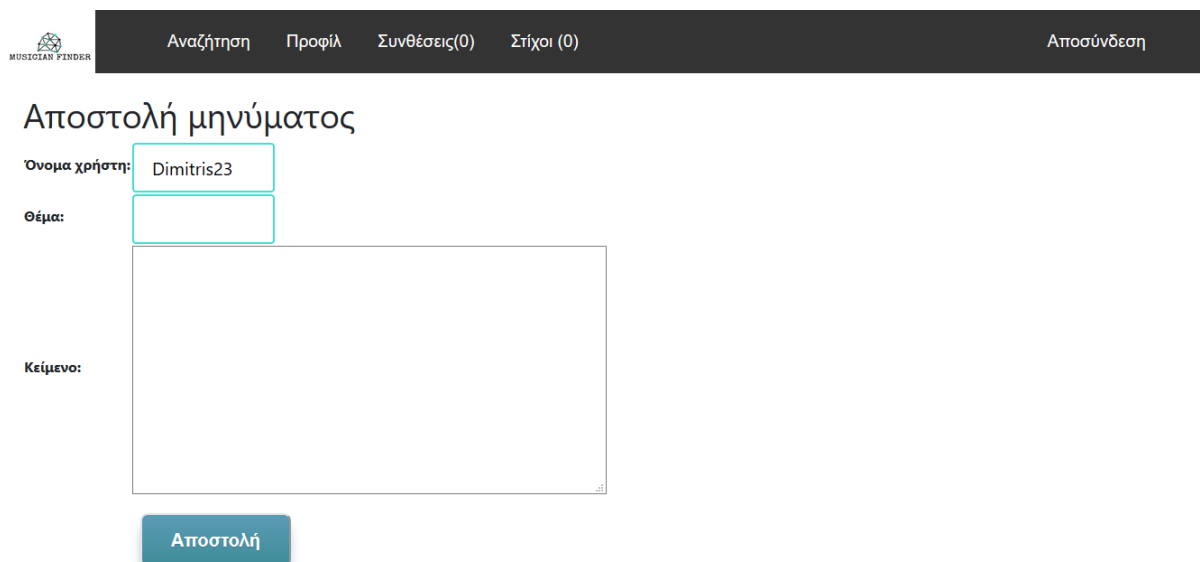
Η view συνάρτηση για την παραπάνω λίστα είναι η **other_lyrics_list** και έχει τον εξής κώδικα:

```
@login_required  
def other_lyrics_list(request, prof_id):  
    prof = get_object_or_404(Profile, pk=prof_id)  
    return render(request, 'other_lyrics_list.html', {'profile': prof})
```

Απόσπασμα Παρουσίασης Εφαρμογής 37: other_lyrics_list

4.3.11 Messaging

Από την **other_lyrics_list_page** ο χρήστης επιστρέφει στην **profile_appearance_page** και επιλέγει το link «Στείλε μήνυμα» (βλ 4.3.8). Τότε μεταφέρεται στην **write_message_page** όπου μπορεί να στείλει μήνυμα στον Dimitris23:



Απόσπασμα Παρουσίασης Εφαρμογής 38: write_message

Η παραπάνω σελίδα παρέχεται από την εφαρμογή **django-postman**. Η εφαρμογή αυτή μας επίσης τη δυνατότητα να έχουμε αρχειοθετημένα, εισερχόμενα ,απεσταλμένα και διαγραμμένα μηνύματα βάζοντας απλά τα αντίστοιχα links στην template. Για την συγκεκριμένη εφαρμογή υπάρχει αναλυτικά η λειτουργικότητά της σε [documentation\(πηγή : https://buildmedia.readthedocs.org/media/pdf/django-postman/latest/django-postman.pdf\)](https://buildmedia.readthedocs.org/media/pdf/django-postman/latest/django-postman.pdf)

) . Επίσης μπορούμε να επεξεργαστούμε τα default templates και views της εφαρμογής αυτής. Στην περίπτωση μας έχουμε επεξεργαστεί τα default templates της ώστε να ταιριάζουν σε κάποιο βαθμό με την υπόλοιπη εφαρμογή.

Για να στείλει μήνυμα ο χρήστης πρέπει να συμπληρώσει υποχρεωτικά και τα τρία πεδία. Το πεδίο «Όνομα χρήστη», όποτε επισκεπτόμαστε την σελίδα που είναι για την αποστολή μηνύματος μέσω κάποιου προφίλ από τον σύνδεσμο «Στείλε μήνυμα», συμπληρώνεται εκ των προτέρων.

Στο παράδειγμά μας ο χρήστης συμπληρώνει τα πεδία ως εξής:

Αποστολή μηνύματος

Όνομα χρήστη:

Dimitris23

Θέμα:

Συνεργασία

Κείμενο:

Καλησπέρα Dimitris23! Θα ήθελα να βρεθούμε για συνεργασία! Πότε μπορείς και που?

Αποστολή

Απόσπασμα Παρουσίασης Εφαρμογής 39: message_before_sending

Μετά πατά το link «Αποστολή» και γίνεται redirect στην **profile_appearance_page** του Dimitris23 όπου φαίνεται και το μήνυμα επιβεβαίωσης της αποστολής του μηνύματος:

The screenshot shows the profile page of a user named Dimitris23. At the top, there is a navigation bar with links: Αναζήτηση, Προφίλ, Συνθέσεις(0), Στίχοι (0), and Αποσύνδεση. The profile section includes a placeholder for a profile picture, the username Dimitris23, and the email address dimitri23@gmail.com. Below the email, there is a link 'Στείλτε μήνυμα' and the text 'Το μήνυμα εστάλη'. The profile also lists musical interests: 'Μουσικές ιδιότητες: Τραγουδιστής, Κιθαρίστας' and 'Μουσικά ακούσματα: Rock: Classic Rock, Psychedelic Rock, Blues: Country Blues'.

Απόσπασμα Παρουσίασης Εφαρμογής 40: message_sent

Βλέποντας από το προφίλ του Dimitris23, επιλέγουμε το link «Εισερχόμενα μηνύματα» από την **profile_page** του και μεταφερόμαστε στην **inbox_page** όπου μπορούμε να δούμε το μήνυμα που τους στείλαμε

Εισερχόμενα Μηνύματα

Συζητήσεις [Μηνύματα](#)

Ενέργεια [Αποστολέας Θέμα](#)

George2019 [Συνεργασία](#) - Καλησπέρα! Θα ήθελα να βρεθούμε για συνεργασία! Πότε μπορείς και που? 10:18 AM

[Διαγραφή](#)
[Σημειώστε ως διαβασμένο](#)
[Σημειώστε ως μη διαβασμένο](#)
[Παραλείφθηκε](#)

Απόσπασμα Παρουσίασης Εφαρμογής 41: inbox

Στην παραπάνω σελίδα βλέπουμε πληροφορίες όπως το username του αποστολέα, το θέμα, μια πρόταση από το μήνυμα και τότε αυτό παραλείφθηκε. Επίσης μπορούμε να διαγράψουμε το μήνυμα, να το σημειώσουμε ως διαβασμένο ή ως μη διαβασμένο και να κατηγοριοποιήσουμε τα μηνύματα σε απλά μηνύματα ή σε συζητήσεις.

Για να διαβάσουμε ολόκληρο το μήνυμα και να απαντήσουμε, πατάμε το Link με την ονομασία του θέματος. Στην περίπτωσή μας, πατάμε το link «Συνεργασία». Τότε μεταφερόμαστε την **view_message_page** όπου βλέπουμε το μήνυμα και μπορούμε να γράψουμε κείμενο απάντησης και να το στείλουμε:

Μήνυμα

George2019 » <me> | Sept. 24, 2019, 10:18 a.m. | Συνεργασία
Καλησπέρα! Θα ήθελα να βρεθούμε για συνεργασία! Πότε μπορείς και που?
[Επιστροφή](#) [Διαγραφή](#) [Απάντηση](#)

Απάντηση

Καλησπέρα! Αύριο στις 9 στο small8 βολεύει?

[Απάντηση](#)

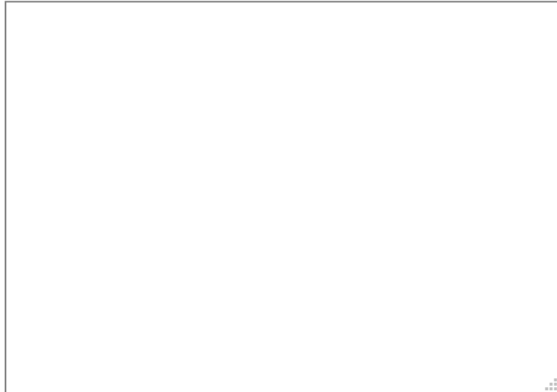
Απόσπασμα Παρουσίασης Εφαρμογής 41: answer

Αφού πατήσουμε το link «Απάντηση», επιστρέφουμε στην **inbox_page** και αν τώρα πατήσουμε το Link «Συνεργασία» μπορούμε να δούμε την συζήτηση μέχρι τώρα :

Συζήτηση

George2019 » <me> | Sept. 24, 2019, 10:18 a.m. | Συνεργασία
Καλησπέρα! Θα ήθελα να βρεθούμε για συνεργασία! Πότε μπορείς και που?
<me> » George2019 | Sept. 24, 2019, 10:24 a.m. | Re: Συνεργασία
Καλησπέρα! Αύριο στις 9 στο small8 βολεύει?
[Επιστροφή](#) [Διαγραφή](#) [Απάντηση](#)

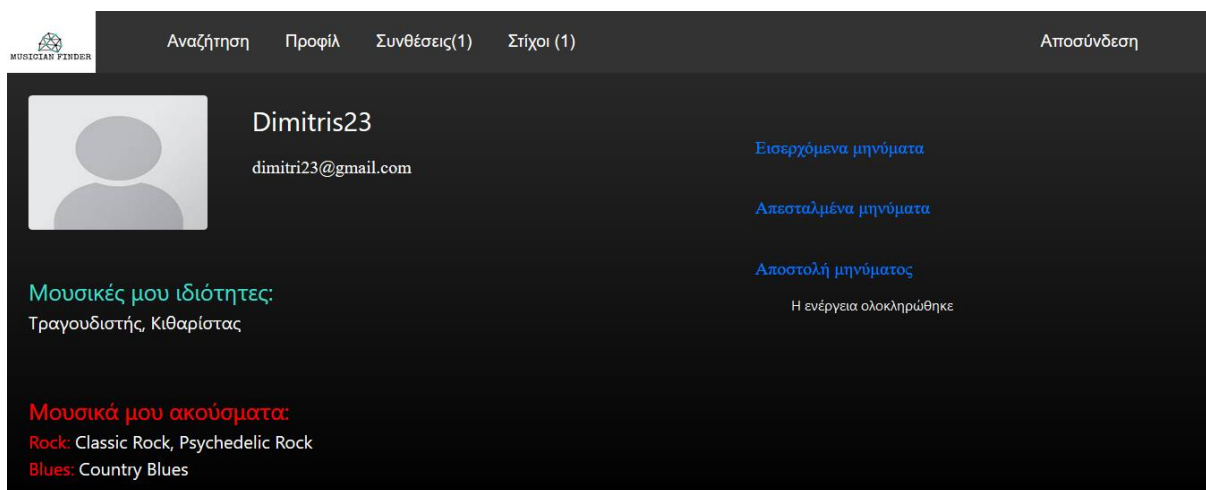
Απάντηση



Απάντηση

Απόσπασμα Παρουσίασης Εφαρμογής 41: conversation

Μετά επιστρέφουμε στην **profile_page** του Dimitris23 όπου φαίνεται μήνυμα επιβεβαίωσης της παραπάνω ενέργειας:



Απόσπασμα Παρουσίασης Εφαρμογής 42: message_sent_completed

4.3.12 Logout/Login

Συνεχίζοντας με το προφίλ του Dimitris23 πατάμε από το navigation bar το link «Αποσύνδεση». Τότε γίνεται redirect στην **login_page**, όπως έχει οριστεί στο **settings.py**:

```
LOGOUT_REDIRECT_URL = 'login'
```

Τότε μεταφερόμαστε στην **login_page**:

Απόσπασμα Παρουσίασης Εφαρμογής 43: login_page

Για να γίνει το παραπάνω redirect χρησιμοποιούνται δυο url patterns. Το ένα «γεφυρώνει» το url “accounts” με την εφαρμογή με τα urls της εφαρμογής django.contrib.auth:

```
path('accounts/', include('django.contrib.auth.urls'), name='accounts'),
```

Το άλλο url pattern βρίσκεται μέσα στην εφαρμογή django.contrib.auth και αναφερόμαστε σε αυτό με το namespace ‘login’. Η view συνάρτηση που καλείται μέσω αυτού ψάχνει αυτόματα να βρει template που να ονομάζεται **login.html** σε υποφάκελο του φακέλου “templates”, που ονομάζεται “registration”. Αν συμπληρωθούν σωστά τα στοιχεία μας τότε μεταφερόμαστε στην **home_page**.

ΚΕΦΑΛΑΙΟ 5

5.1 Συμπεράσματα και μελλοντικές επεκτάσεις

Ολοκληρώνοντας την υλοποίηση της εφαρμογής μου συνειδητοποίησα πως η ανάπτυξη μια διαδικτυακής εφαρμογής μέσω ενός web framework

είναι η γρηγορότερη και κυρίως η ασφαλέστερη διαδικασία. Βέβαια είναι απαραίτητο να γνωρίζουμε σε καλό επίπεδο την γλώσσα προγραμματισμού που βασίζεται το framework που θα χρησιμοποιήσουμε διότι πολλά tutorials και documentations λαμβάνουν ως δεδομένες κάποιες προγραμματιστικές γνώσεις πάνω στη γλώσσα αυτή. Επίσης απέκτησα μεγαλύτερη αντίληψη σχετικά με το πώς δουλεύουν οι σύγχρονες διαδικτυακές εφαρμογές διότι οι δυσκολίες που συναντούσα κατά την ανάπτυξης της δικής μου διαδικτυακής εφαρμογής με έβαλαν στη διαδικασία να κατανοήσω καλύτερα την λειτουργικότητα αυτών.

Είναι βέβαιο ότι η συγκεκριμένη εφαρμογή στο στάδιο που βρίσκεται αφήνει χώρο για αρκετές επεκτάσεις στο μέλλον. Κάποια από αυτές που σκοπεύω να υλοποιήσω η λειτουργία της εφαρμογής σε production mode. Η δυνατότητα εισαγωγής σχολίων και κριτικών στις συνθέσεις και στους στίχους ώστε να γίνεται καλύτερα η αξιολόγηση ενός μουσικού με βάση την γνώμη άλλων μουσικών. Η δυνατότητα συνεργασίας για την υλοποίηση μουσικού κομματιού online. Η μετάφραση όλων των σελίδων σε διάφορες γλώσσες ώστε να μεγαλώσει η κοινότητα που συμμετέχει στην εφαρμογή. Η προσαρμογή της εφαρμογής σε όλους τους browsers και στα κινητά.

ΚΕΦΑΛΑΙΟ 6

6.1 Δικτυογραφία

1. <https://en.wikipedia.org/wiki/Framework>
2. https://en.m.wikipedia.org/wiki/Web_framework
3. https://en.m.wikipedia.org/wiki/Software_framework
4. <https://www.techopedia.com/definition/14384/software-framework>
5. <https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/>
6. https://en.wikipedia.org/wiki/Business_logic
7. https://linuxhint.com/10_best_math_libraries_python/
8. <https://djangostars.com/blog/what-is-a-web-framework/>
9. <https://www.keycdn.com/blog/best-backend-frameworks>
10. <https://book.cakephp.org/2.0/en/core-libraries.html>
11. <http://jinja.pocoo.org/docs/2.10/>
12. <https://searchmicroservices.techtarget.com/definition/RESTful-API>
13. https://en.wikipedia.org/wiki/Secure_cookie
14. https://en.wikipedia.org/wiki/Unit_testing
15. <https://flask-restful.readthedocs.io/en/0.3.5/quickstart.html>
16. https://setuptools.readthedocs.io/en/latest/easy_install.html

17. <https://www.webforefront.com/django/configuredjangosettings.html>
18. https://www.quora.com/What-is-the-SECRET_KEY-in-Django-actually-used-for
19. https://en.wikipedia.org/wiki/Cryptographic_hash_function
20. <https://docs.djangoproject.com/en/2.2/ref/settings/>
21. <https://docs.djangoproject.com/en/2.2/topics/auth/>
22. https://en.wikipedia.org/wiki/Software_framework