ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Ανάπτυξη διαδικτυακής εφαρμογής για την διαχείριση ακαδημαϊκών ημερολογίων

Πτυχιακή εργασία

Κοβάτση Ευδοκία

Αθήνα, 2020

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Καμαλάκης Θωμάς
Αναπληρωτής Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Μιχαλακέλης Χρήστος
Επίκουρος Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Αναγνωστόπουλος Δημοσθένης
Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Η Κοβάτση Ευδοκία

δηλώνω υπεύθυνα ότι:

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

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

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου για την παρούσα εργασία, κ. Καμαλάκη για την καθοδήγηση του κατά τη διάρκεια της ολοκλήρωσης της, αλλά και επειδή ήταν πάντα άμεσα διαθέσιμος για να μου λύσει τυχών απορίες που προέκυψαν. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε σε όλη τη διάρκεια των σπουδών μου.

Πίνακας περιεχομένων

Περίληψη

Σκοπός αυτής της πτυχιακής είναι η διαχείριση των ακαδημαϊκών ημερολογίων με τη δημιουργία μιας εφαρμογής που θα επιτρέπει στο χρήστη να εισάγει μαζικά συμβάντα στο Google Calendar. Δίνει επίσης τη δυνατότητα στο χρήστη να δημιουργήσει ένα CSV αρχείο με τα συμβάντα μεταξύ δύο ημερομηνιών.

Η εφαρμογή είναι υλοποιημένη σε περιβάλλον Django, το οποίο χρησιμοποιεί τη γλώσσα προγραμματισμού Python. Αφού ανεβάσει το αρχείο CSV ο χρήστης, η εφαρμογή διαβάζει το περιεχόμενο και με τη βοήθεια του Google Calendar API μπορεί και επικοινωνεί με το Google Calendar, ώστε να δημιουργήσει τα συμβάντα που θέλει ο χρήστης. Το Google Calendar API επιτρέπει τη δημιουργία συμβάντος με συγκεκριμένη ημερομηνία εκπλήρωσης αλλά και τη δημιουργία ενός που επαναλαμβάνεται. Η εφαρμογή επιτρέπει επίσης στο χρήστη να προσθέσει κάποιες παραπάνω πληροφορίες στο συμβάν όπως για παράδειγμα μια περιγραφή ή την τοποθεσία που θα γίνει.

Στην εργασία περιλαμβάνονται περιγραφές για την έννοια και τα στοιχεία του Google Calendar API, πως υλοποιείται η Python σε περιβάλλον Django, η διαδικασία υλοποίησης και ο κώδικας της εφαρμογής, καθώς και εικόνες από τη τελική εμφάνιση και τη χρήση της εφαρμογής.

Λέξεις κλειδιά: [Ανάπτυξη Διαδικτυακής Εφαρμογής Ημερολογίου, Προγραμματιστική Διεπαφή, Μαζική Δημιουργία Συμβάντων]

Abstract

This paper is about the creation of an application that can add multiple events in a Google Calendar. More specifically, the user uploads a CSV file from their computer that has all the events they wish to create and chooses the calendar they want to add them to. The application also allows the user to download a CSV file with the events between two dates from their Google Calendar.

The application is created in a Django environment, using the Python computing language. After the user uploads their CSV file, and picks in which calendar they wish to add their events to, the application reads the contents and using the Google Calendar API, it adds the events from the file to the selected Google Calendar. The Google Calendar API allows the creation of events that happen on a specific date or recurring events. The application also allows the user to add some extra information for the events they want to create, for example a description or the location the event will take place.

In later chapters I will describe more about the Google Calendar API, how to implement Python in a Django environment, the process of writing the code and creating the application, and include screen shots from the final product.

Keywords: [Web Application for Google Calendar, API, Mass Creation of Events]

Κατάλογος Εικόνων

Συντομογραφίες

API	Application programming interface
CSV	Comma Separated Values
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
ICAL	iCalendar
URL	Uniform Resource Locator
URI	Uniform Resource Identifier

Εισαγωγή

Στη σημερινή εποχή που οι άνθρωποι έχουν γεμάτα προγράμματα και πολλές υποχρεώσεις, οι περισσότεροι επικαλούνται τη βοήθεια online ημερολογίων για να βάλουν σε μια τάξη τις υποχρεώσεις της ημέρας. Ένα από τα πιο γνωστά online ημερολόγια είναι αυτό της Google. Η χρήση του είναι πολύ διαδεδομένη λόγω του ότι προσφέρετε δωρεάν σε όλους τους χρήστες που διαθέτουν ένα λογαριασμό Gmail, καθώς και λόγω του ότι διαθέτει και μια εύχρηστη εφαρμογή για κινητά δίνοντας πρόσβαση στους χρήστες του όπου και να βρίσκονται.

Η πτυχιακή αυτή πραγματεύεται τη δημιουργία μιας εφαρμογής για τη μαζική εισαγωγή συμβάντων στο ημερολόγιο της Google από ένα αρχείο CSV. Ο χρήστης συμπληρώνει σε ένα αρχείο CSV όλα τα συμβάντα που θέλει να εισάγει στο ημερολόγιό του και η εφαρμογή τα προσθέτει στο Google Calendar της επιλογής του χρήστη. Το CSV αρχείο έχει κάποια συγκεκριμένα κελιά για το κάθε συμβάν τα οποία πρέπει να συμπληρώσει ο χρήστης. Κάποια από αυτά είναι υποχρεωτικά, όπως η ημερομηνία και ώρα έναρξης και λήξης του συμβάντος, ενώ κάποια άλλα είναι απλώς παραπάνω πληροφορίες που μπορεί να συμπληρώσει ο χρήστης αν αυτός επιθυμεί, όπως ποιοι είναι οι συμμετάσχοντες και η τοποθεσία του συμβάντος.

Ο λόγος ανάγκης δημιουργίας μιας τέτοιας εφαρμογής είναι κυρίως η εισαγωγή πολλαπλών συμβάντων που παίρνουν μέρος μέσα σε ένα μεγάλο χρονικό διάστημα. Όταν προσθέτουμε ένα συμβάν στο Google Calendar απευθείας πρέπει να κάνουμε κλικ στο κουμπί ‘δημιουργία’, να μεταβούμε στην επιθυμητή ημερομηνία ή να την ψάξουμε στη μικρογραφία ημερολογίου που υπάρχει στην αριστερή μεριά της σελίδας και μετά να συμπληρώσουμε όλα τα υπόλοιπα στοιχεία που θέλουμε. Μπορεί για κάποιους να μην ακούγεται τόσο χρονοβόρο αλλά για σκεφτείτε μια εταιρεία που θέλει να προσθέσει όλα τα meeting του έτους, μια ναυτιλιακή εταιρεία που θέλει να προσθέσει τους ελέγχους των πλοίων της για τους επόμενους 3 μήνες ή ένα πανεπιστήμιο που θέλει να προσθέσει όλα τα μαθήματα του εξαμήνου. Με την εφαρμογή αυτή ο χρήστης γλιτώνει τα επαναλαμβανόμενα κλικ για τη

δημιουργία και την αποθήκευση των συμβάντων, καθώς επίσης του δίνεται και η δυνατότητα να χρησιμοποιήσει οποιεσδήποτε συντομεύσεις του παρέχει ένα πρόγραμμα δημιουργίας αρχείων CSV όπως η μαζική αντιγραφή και επικόλληση.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το framework Django, ένα ελαφρύ framework με μεγάλη ικανότητα επέκτασης. Το Django χρησιμοποιεί τη γλώσσα προγραμματισμού Python για το back end και την HTML για το front end. Η Python είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού που χρησιμοποιείται σε χιλιάδες εφαρμογές στη σημερινή εποχή λόγω της ευχρηστίας της και της ευκολίας στη σύνταξή της. Η HTML είναι η στάνταρ γλώσσα για αρχεία που είναι σχεδιασμένα να ανοιχτούν από έναν browser. Μπορεί να βοηθηθεί από τεχνολογίες όπως τη γλώσσα CSS για να γίνει πιο φιλική προς το χρήστη και τη γλώσσα JavaScript για να φέρει εις πέρας πιο περίπλοκες διαδικασίες. Η επικοινωνία μεταξύ της εφαρμογής και του Google Calendar ώστε να μπορέσει να δημιουργήσει τα συμβάντα που επιθυμεί ο χρήστης γίνεται μέσω του Google Calendar API, ένα REST API που παρέχεται από την Google και επιτρέπει στην εφαρμογή πρόσβαση στα χαρακτηριστικά του Google Calendar.

Στο κεφάλαιο 1 αναλύονται παραπάνω οι τεχνολογίες και τα λογισμικά που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής. Στο κεφάλαιο 2 παρουσιάζονται πληροφορίες για την αρχιτεκτονική του συστήματος. Στο κεφάλαιο 3 υπάρχουν τα βήματα που ακολουθήθηκαν για την εγκατάσταση των τεχνολογιών και εργαλείων που χρησιμοποιήθηκαν καθώς και εικόνες από σημεία του κώδικα που αξίζουν να σημειωθούν. Στο κεφάλαιο 4 μπορούμε να βρούμε εικόνες από την εκτέλεση της εφαρμογής. Τέλος, στο κεφάλαιο 5 είναι τα συμπεράσματα και προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής.

1. Τεχνολογίες και λογισμικά υλοποίησης

Σε αυτό το κεφάλαιο αναφέρονται τα λογισμικά και εργαλεία που χρησιμοποιήθηκαν για τη δημιουργία της συγκεκριμένης εφαρμογής. Πριν από αυτό όμως αξίζει να αναφερθεί ότι υπάρχει τρόπος να προσθέσει κάποιος μαζικά συμβάντα στο Google Calendar. Το ίδιο το Google Calendar προσφέρει τη δυνατότητα να δημιουργήσει ένας χρήστης πολλαπλά συμβάντα μέσω ενός αρχείου CSV ή iCal.

Το Google Calendar επιτρέπει την εισαγωγή συμβάντων από ένα αρχείο CSV που έχει τα εξής πεδία: Subject (το όνομα του συμβάντος), Start Date (Η ημερομηνία που ξεκινάει το συμβάν), Start Time (Η ώρα που ξεκινάει το συμβάν), End Date (Η ημερομηνία που τελειώνει το συμβάν), End Time (Η ώρα που τελειώνει το συμβάν), All Day Event (Αν το συμβάν διαρκεί ολόκληρη τη μέρα με τιμές True ή False), Description (Η περιγραφή του συμβάντος), Location (Η τοποθεσία του), Private (Αν το συμβάν είναι προσωπικό ή δημόσιο). Όπως παρατηρούμε λείπει ένα βασικό χαρακτηριστικό για την αναμενόμενη χρήση του από τη γραμματεία, η αυτόματη εισαγωγή επαναλαμβανόμενων συμβάντων. Αν θέλουμε να δημιουργήσουμε επαναλαμβανόμενα συμβάντα με αυτό το τρόπο πρέπει να προσθέσουμε κάθε διαφορετική περίσταση του συμβάντος ξεχωριστά στο CSV αρχείο. Αξίζει λοιπόν η δημιουργία μιας εφαρμογής που μπορεί επίσης να δημιουργήσει επαναλαμβανόμενα συμβάντα.

Εκτός από την εισαγωγή μέσω αρχείου CSV, το Google Calendar επιτρέπει την εισαγωγή συμβάντων από ένα αρχείο iCal. Τα αρχεία iCal είναι αρχεία ημερολογίου που χρησιμοποιούνται από πολλαπλές εφαρμογές email και ημερολογίων, όπως το Microsoft Outlook, το Google Calendar, και το Apple Calendar. Δίνει τη δυνατότητα στο χρήστη να δημοσιεύσει ή να μοιραστεί πληροφορίες για ένα ημερολόγιο στο διαδίκτυο ή μέσω email. Συνήθως χρησιμοποιούντα για να στείλει ένας χρήστης αίτημα σύσκεψης σε άλλους χρήστες που μπορούν να εισάγουν το αρχείο στο δικό τους ημερολόγιο. Λόγω του μη σύνηθες format και της προϋπόθεσης ότι προέρχεται από ένα ήδη υπάρχοντα ημερολόγιο, δε μας χρησιμεύει στο συγκεκριμένο πρόβλημα.

1.1. Django

Το Django είναι ένα ένα δωρεάν και ανοιχτού κώδικα web framework βασισμένο στην γλώσσα προγραμματισμού Python. Ακολουθεί την αρχιτεκτονική model-template-view (MTV) η οποία δουλεύει παρόμοια με την αρχιτεκτονική MVC απλά το Django αποκαλεί τα HTTP αντικείμενα Views και τα αρχεία HTML τα ονομάζει Templates. Ο κύριος στόχος του Django είναι η ευκολία στη δημιουργία σύνθετων ιστοσελίδων με βάσεις δεδομένων που έχουν τη δυνατότητα να μεγαλώσουν. Το Django δίνει έμφαση στη δυνατότητα να επαναχρησιμοποιήσουμε ότι φτιάχνουμε και στο "pluggability" των components που δημιουργούμε. Αυτό σημαίνει ότι μπορούμε εύκολα να χρησιμοποιήσουμε προηγούμενα project ή να ενσωματώσουμε project άλλων για να μη χρειαστεί να γράψουμε τα πάντα από την αρχή.

Το Django παρέχει ένα ανεξάρτητο web server για τη δημιουργία και τη διαδικασία testing του project μειώνοντας σημαντικά το χρόνο και την ταλαιπωρία που χρειάζεται κάποιος για να ξεκινήσει να γράφει και να δοκιμάζει των κώδικα της εφαρμογής. Διαθέτει επίσης ένα σύστημα για την εύκολη δημιουργία forms καθώς και τον αυτοματοποιημένο έλεγχο των πεδίων που συμπληρώνει ο χρήστης μειώνοντας την επανάληψη κώδικα ελέγχου. Τέλος, υπάρχει η δυνατότητα κληρονομικότητας και στα Templates, δηλαδή τα αρχεία που διαθέτουν τον κώδικα HTML που καθορίζουν πως θα φαίνεται η ιστοσελίδα μας στο χρήστη. Αυτό βοηθάει στην ενσωμάτωση ενός καθολικού θέματος σε όλες τις σελίδες της εφαρμογής μας χωρίς την επανάληψη κώδικα.

1.2. Python

Η Python χρησιμοποιείται παντού στο Django, ακόμα και στο αρχείο που καθορίζει τα settings. Η Python είναι μια διερμηνευόμενη, γενικού σκοπού και υψηλού επιπέδου δυναμική γλώσσα προγραμματισμού. Ας αναλύσουμε λίγο παραπάνω τι σημαίνει ο παραπάνω ορισμός.

Διερμηνευόμενη: Στον προγραμματισμό, διερμηνευόμενη γλώσσα είναι μια γλώσσα προγραμματισμού η υλοποίηση της οποίας συνήθως αποτελείται από έναν διερμηνέα. Θεωρητικά, οποιαδήποτε γλώσσα μπορεί να είναι είτε μεταγλωττισμένη είτε διερμηνευμένη, έτσι ο διαχωρισμός αυτός εφαρμόζεται μόνο με βάση την συνήθη πρακτική υλοποίησης και όχι κάποια συγκεκριμένη ιδιότητα μιας γλώσσας.

Γενικού σκοπού: Στον προγραμματισμό, μια γενικού σκοπού γλώσσα προγραμματισμού είναι μια γλώσσα σχεδιασμένη για τη γραφή λογισμικού σε ευρύτερους τομείς. Λέγεται έτσι γιατί δε διαθέτει constructs σχεδιασμένα για να χρησιμοποιηθούν σε συγκεκριμένους τομείς. Για να κατανοήσουμε καλύτερα τι σημαίνει αυτό αρκεί να δούμε τι σημαίνει όταν μια γλώσσα προγραμματισμού είναι ειδικού σκοπού. Οι γλώσσες ειδικού σκοπού (ή Domain-specific languages ή DSL) είναι γλώσσες που ειδικεύονται σε συγκεκριμένους τομείς όπως η HTML που χρησιμοποιείται για τις ιστοσελίδες και η SQL για τη συνομιλία με βάσεις δεδομένων.

Υψηλού επιπέδου: Ως υψηλού επιπέδου γλώσσα προγραμματισμού ορίζεται αυτή που επιτρέπει τη μεταφορά ενός προγράμματος από έναν υπολογιστή σε έναν άλλο. Αποτελείται από εντολές εύκολα κατανοητές στον προγραμματιστή, καθώς μοιάζουν με περιορισμένη φυσική γλώσσα.

Δυναμική γλώσσα προγραμματισμού: Ο όρος δυναμική γλώσσα προγραμματισμού χρησιμοποιείται ευρύτατα στην επιστήμη των υπολογιστών για να περιγράψει γλώσσες προγραμματισμού υψηλού επιπέδου που παρουσιάζουν κατά το χρόνο εκτέλεσης συμπεριφορά που άλλες γλώσσες παρουσιάζουν κατά τη μετάφραση. Αυτές οι συμπεριφορές μπορεί να αποτελούν επέκταση του προγράμματος, είτε προσθέτοντας νέο κώδικα, είτε επεκτείνοντας αντικείμενα και ορισμούς, είτε τροποποιώντας το σύστημα τύπων, όλα κατά τη διάρκεια της εκτέλεσης του προγράμματος.

1.2.1. **Χαρακτηριστικά της Python**

Η Python είναι εύκολη στην εκμάθηση λόγω της ομοιότητας της με την αγγλική γλώσσα και της ικανότητάς της να γράφει κανείς προγράμματα με λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες προγραμματισμού. Μπορεί να τρέξει σε διάφορα συστήματα (Windows, Mac, Linux, etc.) και η σύνταξή της είναι πιο 'φιλική' προς τον άνθρωπο λόγο του ότι κάθε καινούρια γραμμή είναι μια εντολή και κάθε block κώδικα, όπως ένα if ή ένα for υποδεικνύεται γράφοντας τον κώδικα του block ένα tab πιο μέσα σε αντίθεση με άλλες γλώσσες προγραμματισμού που συνήθως χρησιμοποιούν την αγκύλη ({ })

1.3. Atom

Το Django δημιουργεί απλά το directory της εφαρμογής και δε διαθέτει text editor για τη γραφή και επεξεργασία του κώδικα, επιτρέποντας έτσι την επιλογή του από το χρήστη με βάση τις προτιμήσεις του. Για την υλοποίηση της συγκεκριμένης πτυχιακής χρησιμοποίησα το εργαλείο Atom. Το Atom είναι ένας δωρεάν και ανοιχτού κώδικα text editor που διαθέτει syntax highlighting για διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένου της Python και της HTML που χρησιμοποιήθηκαν για τη σύνταξη της εφαρμογής. Το syntax highlighting είναι ένα feature που διαθέτουν οι text editors που χρησιμοποιούνται για τη γραφή κώδικα, το οποίο αλλάζει το χρώμα και τη γραμματοσειρά σε σημεία του κώδικα όπως τη πρώτη γραμμή κάθε μεθόδου ώστε να είναι πιο ευανάγνωστο. Το Atom διαθέτει επίσης την ικανότητα να προβάλει το directory στο ίδιο περιβάλλον που γράφεται ο κώδικας, πράγμα που κάνει την αλλαγή μεταξύ αρχείων κατά τη διάρκεια του γραψίματος πολύ πιο γρήγορη.

1.4. Bootstrap

Το Bootstrap είναι ένα ανοιχτού κώδικα toolkit για την ανάπτυξη κώδικα HTML, CSS και Javascript. Για τη συγκεκριμένη πτυχιακή χρησιμοποιήθηκε ως stylesheet για τη μετατροπή των ιστοσελίδων σε πιο φιλικές προς το χρήστη. Περιλαμβάνει έτοιμα στυλ CSS που χρησιμεύουν στη τοποθέτηση των κάθε στοιχείων βασισμένα στο flexbox, καθώς και στυλ για γραμματοσειρές, κουμπιά και φόρμες.

1.5. Google Calendar API

Φυσικά για την αλληλεπίδραση της εφαρμογής με το Google Calendar χρησιμοποιήθηκε το αντίστοιχο API της Google. Το API είναι ένας τρόπος να αλληλεπιδρούν εφαρμογές μεταξύ τους χωρίς να χρειάζεται να δουν η μία τον κώδικα ή τα components της άλλης.

Το Google Calendar API είναι ένα REST API το οποίο είναι προσβάσιμο μέσω HTTP calls και εκθέτει τα περισσότερα feature που είναι διαθέσιμα μέσω της διεπαφής του Google Calendar. Ο κάθε χρήστης που έχει λογαριασμό Google, διαθέτει ένα κύριο ημερολόγιο που ονομάζεται primary calendar το οποίο δημιουργείται αυτόματα όταν ο χρήστης αποκτά το λογαριασμό Google. Πέραν αυτού ο χρήστης μπορεί έπειτα να δημιουργήσει άλλα δευτερεύοντα ημερολόγια, το καθένα με το δικό του όνομα. Το Google Calendar API μας επιτρέπει να έχουμε πρόσβαση σε αυτά τα ημερολόγια για να τα διαβάσουμε ή να τα τροποποιήσουμε, καθώς και να προσθέσουμε, να διαγράψουμε ή να τροποποιήσουμε συμβάντα που ανήκουν σε αυτά εφόσον μας το έχει επιτρέψει ο χρήστης. Περισσότερα για το Google Calendar API, την εκτέλεση των κλήσεων σε αυτό, καθώς και περισσότερες λεπτομέρειες για τις δυνατότητές του σε επόμενο κεφάλαιο.

2. Αρχιτεκτονική Συστήματος

Σκοπός αυτού του κεφαλαίου είναι η ανάλυση και η περιγραφή της αρχιτεκτονικής η οποία ακολουθεί η εφαρμογή όπως επίσης και η λειτουργικότητα κάθε συστατικού που αλληλεπιδρά με το χρήστη.

2.1. Περιγραφή Αρχιτεκτονικής

Το σύστημα που δημιουργήθηκε στα πλαίσια αυτής της πτυχιακής αποτελείται από τρία τμήματα. Αρχικά υπάρχει το interface που έρχεται σε επαφή ο χρήστης. Οι ιστοσελίδες της εφαρμογής δίνουν τη δυνατότητα στο χρήστη να ανεβάσει το CSV αρχείο που επιθυμεί, να κατεβάσει ένα CSV αρχείο με τα συμβάντα μεταξύ δύο ημερομηνιών που θα επιλέξει, καθώς και μια σελίδα με οδηγίες για την ομαλή ολοκλήρωση των δύο αυτών διαδικασιών. Υπάρχει επίσης μια σελίδα που βλέπει ο χρήστης πριν κάνει Log in που εξηγεί με λίγα λόγια τις δυνατότητες της εφαρμογής.

Το δεύτερο τμήμα είναι το Google Calendar API. Το Google Calendar API μας επιτρέπει, αφού ζητήσουμε άδεια από το χρήστη, να έχουμε πρόσβαση στα διάφορα ημερολόγια του λογαριασμού του και να μπορούμε να διαβάσουμε και να δημιουργήσουμε καινούρια συμβάντα σε αυτά.

Το τρίτο και τελευταίο τμήμα είναι το κομμάτι της εφαρμογής που ενώνει τα δύο αυτά τμήματα και εκτελεί όλη την απαραίτητη επεξεργασία για τη μετατροπή των δεδομένων από ένα αρχείο CSV σε μορφή κατάλληλη για το Google Calendar API και το αντίστροφο.

Πάμε να δούμε μια πιο αναλυτική περιγραφή για το κάθε τμήμα που αναφέρθηκε παραπάνω.

2.2. Τμήμα επαφής με το χρήστη

Το τμήμα επαφής με το χρήστη είναι οι σελίδες της εφαρμογής που βλέπει ο χρήστης στο browser καθώς και όλα τα συστατικά στοιχεία τους. Συγκεκριμένα το site αποτελείται από 4 σελίδες. Κοινό σε όλες τις σελίδες είναι το navigation menu στο πάνω μέρος ώστε να μπορεί να βρει ο χρήστης τη σελίδα που επιθυμεί.

2.2.1. Before Log in

Για να μπορέσει η εφαρμογή με τη βοήθεια του Google Calendar API να εμφανίσει μια λίστα με τα ημερολόγια του χρήστη ώστε να μπορέσει ο χρήστης να διαλέξει και να προσθέσει τα συμβάντα στο ημερολόγιο που θα επιλέξει χρειάζεται κάποια δεδομένα του χρήστη. Αυτά τα δεδομένα τα παίρνουμε όταν ο χρήστης κάνει Log in με ένα λογαριασμό Google. Γι' αυτό υπάρχει αυτή η σελίδα. Είναι η πρώτη σελίδα που συναντά ο χρήστης, η οποία του μεταδίδει αυτή την πληροφορία, καθώς και μερικά λόγια για το τι κάνει η συγκεκριμένη εφαρμογή.

2.2.2. Import CSV

Σε αυτή τη σελίδα ο χρήστης μπορεί να ανεβάσει το CSV αρχείο με τα συμβάντα που θέλει να προσθέσει στο Google Calendar. Στο πάνω μέρος της σελίδας βρίσκεται μια φόρμα που χρησιμοποιείται για αυτή τη διαδικασία.

Η φόρμα αποτελείται από: Ένα κουμπί για να επιλέξει ο χρήστης ένα αρχείο CSV από τον υπολογιστή του με τα συμβάντα που θέλει να προσθέσει. Μια λίστα με τα ημερολόγια του χρήστη για να επιλέξει που θέλει να προστεθούν τα συμβάντα. Ένα κουμπί για να ολοκληρώσει τη διαδικασία.

Κάτω από τη φόρμα υπάρχουν οδηγίες για το πως να ανεβάσει ο χρήστης το αρχείο καθώς και πληροφορίες για εργαλεία που μπορούν να δημιουργήσουν ένα CSV αρχείο.

2.2.3. Export CSV

Αυτή η σελίδα χρησιμοποιείται για να δημιουργήσει ένα CSV αρχείο ο χρήστης με όλα τα συμβάντα μεταξύ δύο ημερομηνιών. Όπως και στη προηγούμενη σελίδα στο πάνω μέρος βρίσκεται η φόρμα που χρησιμοποιείται για αυτή τη διαδικασία.

Η φόρμα αποτελείται από: Μια λίστα για να επιλέξει ο χρήστης από ποιο από τα ημερολόγια του θέλει να πάρει τα συμβάντα. Δύο πεδία που μπορεί αν επιλέξει τις ημερομηνίες από και μέχρι που θέλει να πάρει τα συμβάντα. Ένα κουμπί που ολοκληρώνει τη διαδικασία. Τα πεδία εμφανίζουν ένα ημερολόγιο που μπορεί να επιλέξει την ημερομηνία που επιθυμεί ο χρήστης για να αποτραπούν λάθη λόγω formatting και να διευκολύνει το χρήστη. Τέλος, κάτω από τη φόρμα υπάρχουν πάλι οδηγίες για το πως να χρησιμοποιήσει ο χρήστης το συγκεκριμένο εργαλείο.

2.2.4. Help

Η σελίδα Help δίνει μια εικόνα στο χρήστη για το πως πρέπει να είναι το αρχείο CSV που πρέπει να ανεβάσει αν θέλει να χρησιμοποιήσει τη λειτουργία Import. Υπάρχει ένας πίνακας με το όνομα που πρέπει να έχει κάθε πεδίο καθώς και αναλυτικά τι πρέπει να περιέχει και με ποιο format. Επίσης υπάρχει μια σύντομη περιγραφή για το πως να χρησιμοποιήσει κάποιος την εφαρμογή, αλλά και τι θα συμβεί όταν ολοκληρωθεί η διαδικασία. Η σελίδα αυτή είναι προσβάσιμη χωρίς να είναι απαραίτητο το Log in του χρήστη ώστε να μπορεί να αποφασίσει αν θέλει να χρησιμοποιήσει την εφαρμογή πριν δώσει οποιαδήποτε προσωπικά στοιχεία.

2.3. Google Calendar API

Το άλλο component που χρησιμοποιεί η εφαρμογή είναι το Google Calendar API. Όπως ανέφερα και παραπάνω, το API είναι ένας τρόπος για να επικοινωνεί η εφαρμογή με το Google Calendar του χρήστη χωρίς να έχει πρόσβαση σε ανεπιθύμητες πληροφορίες που

μπορεί ο χρήστης να θέλει να κρατήσει μυστικές ή να πρέπει να μην είναι προσβάσιμες λόγω ασφάλειας.

Για να μπορεί να επικοινωνήσει η εφαρμογή με το Google Calendar API και να κάνει χρήση των δυνατοτήτων του πρέπει αρχικά να καθιερώσουμε μια σύνδεση μεταξύ τους. Επειδή τα δεδομένα που πρόκειται να χειριστούμε είναι προσωπικά και όχι δημόσια η σύνδεση αυτή δε γίνεται μόνο με ένα απλό API key αλλά χρειάζεται να χρησιμοποιήσουμε το πρωτόκολλο OAuth2.

Το OAuth2 είναι η δεύτερη έκδοση του πρωτοκόλλου OAuth. Είναι ένα ανοιχτό πρωτόκολλο επικοινωνίας μεταξύ εφαρμογών το οποίο επιτρέπει σε μια εφαρμογή να κάνει HTTP κλήσεις σε έναν server μιας άλλης εφαρμογής. Στη συγκεκριμένη περίπτωση για την εφαρμογή χρειαζόμαστε εξουσιοδότηση από το χρήστη για να πάρουμε κάποιες πληροφορίες για το ημερολόγιο του στο Google. Το OAuth2 δίνει τη δυνατότητα να ζητήσει η εφαρμογή αυτές τις πληροφορίες από το Google χωρίς να γνωρίζει τον κωδικό του χρήστη.

Συγκεκριμένα, όταν η εφαρμογή θέλει να πάρει κάποια πληροφορία από το λογαριασμό Google του χρήστη, τον στέλνει στο server της Google για να την εξουσιοδοτήσει. Ο χρήστης μεταφέρετε σε ένα ασφαλές περιβάλλον της Google που μπορεί να συνδεθεί με το email και τον κωδικό του και βλέπει μια λίστα με τις άδειες που χρειάζεται η εφαρμογή και μπορεί να επιλέξει αν θέλει να την εξουσιοδοτήσει. Αν πατήσει ναι, ο server της Google στέλνει έναν authorization code στην εφαρμογή τον οποίο μπορεί να χρησιμοποιήσει για να ζητήσει ένα access token. Κάθε φορά που η εφαρμογή θέλει να έχει πρόσβαση σε δεδομένα από το ημερολόγιο του χρήστη ελέγχει το token και αν είναι σωστό επιτρέπει στην εφαρμογή να κάνει την αντίστοιχη κλήση API.

Η λίστα με τις άδειες που βλέπει ο χρήστης όταν πάει να κάνει την εξουσιοδότηση για την εφαρμογή προκύπτει από την ύπαρξη μιας μεταβλητής στον κώδικα της εφαρμογής η οποία περιέχει του σκοπούς χρήσης των δεδομένων του χρήστη. Το Google χρησιμοποιεί αυτούς τους σκοπούς για να εμφανίσει το κατάλληλο μήνυμα όταν ζητάει την άδεια του

χρήστη. Συγκεκριμένα για την εφαρμογή χρειάζεται να ζητηθεί άδεια για να διαβάσει και να γράψει σε ημερολόγια του χρήστη αλλά και να διαβάσει και να γράψει συμβάντα.

Πιο αναλυτικά, αφού η εφαρμογή έχει το access token, αρχικά ζητά από το API να διαβάσει και να επιστρέψει μια λίστα με τα ημερολόγια του χρήστη ώστε να συμπληρώσει τη λίστα στις φόρμες στις σελίδες *Import csv* και *Export csv*. Έτσι ο χρήστης μπορεί να επιλέξει ένα από τα ημερολόγια του για να ολοκληρώσει την αντίστοιχη διαδικασία. Το API έχει τη δυνατότητα να μας επιστρέψει διάφορες πληροφορίες για τα ημερολόγια του χρήστη. Για παράδειγμα το χρώμα του ημερολογίου, την περιγραφή του, το id, το όνομά του ή το time zone που έχει ως προεπιλογή. Για τις ανάγκες της εφαρμογής χρειαζόμαστε μόνο το id και το όνομα του ημερολογίου.

Έπειτα αν θέλουμε να δημιουργήσουμε συμβάντα, στην περίπτωση του Import, καλούμε το API και του δίνουμε τα κατάλληλα δεδομένα για τα συμβάντα. Αν θέλουμε να διαβάσουμε συμβάντα, στην περίπτωση του Export, καλούμε το API και του ζητάμε μια λίστα με τα συμβάντα δίνοντας δύο ημερομηνίες.

Στην περίπτωση που θέλουμε να δημιουργήσουμε συμβάντα πρέπει να φτιάξουμε ένα events resource που να περιέχει τις πληροφορίες που θα να έχουν τα συμβάντα που θα προσθέσουμε. Το API δίνει τη δυνατότητα να προσθέσουμε διάφορες πληροφορίες αλλά στη συγκεκριμένη εφαρμογή χρησιμοποιούμε τα πεδία summary, location, description, start, end, attendees και recurrence. Το πεδίο summary είναι ο τίτλος του συμβάντος, το όνομα που εμφανίζεται στο Google Calendar δηλαδή. Το πεδίο location είναι η τοποθεσία που γίνεται το event, δίχως να χρειάζεται να είναι απαραίτητα τοποθεσία στο χάρτη, μπορεί να είναι απλά “αίθουσα 5”. Στο πεδίο description μπορούμε να προσθέσουμε μια περιγραφή για το συμβάν. Τα πεδία start και end είναι λίστες και περιέχουν 3 πεδία. Το πεδίο date, η ημερομηνία που ξεκινάει ή τελειώνει το συμβάν αντίστοιχα. Το πεδίο dateTime που έχει την ώρα που ξεκινάει ή τελειώνει το συμβάν και το πεδίο timeZone που μπορούμε να σημειώσουμε το timeZone που θέλουμε για να είναι σωστή η ώρα που θα προστεθούν τα συμβάντα. Στο πεδίο attendees μπορούμε να βάλουμε τα email αυτών που θα παρευρεθούν ώστε να μπορούν να λάβουν

notification όταν γίνει το συμβάν. Τέλος στο πεδίο recurrence μπορούμε να βάλουμε μια εντολή που να δηλώνει ότι το συμβάν είναι επαναλαμβανόμενο. Στο CSV αρχείο που ανεβάζει ο χρήστης δηλώνει την ημερομηνία που επιθυμεί να σταματήσει να επαναλαμβάνεται το συμβάν και το API θα κάνει το συμβάν επαναλαμβανόμενο κάθε εβδομάδα την ημέρα που είναι στο πεδίο start μέχρι την ημερομηνία που είναι στο πεδίο recurrence.

Στην περίπτωση που θέλουμε να διαβάσουμε συμβάντα τα πράγματα είναι πιο απλά. Καλούμε το API με το κατάλληλο calendar id και 2 ημερομηνίες. Το πεδίο timeMin δηλώνει την αρχική ημερομηνία που θέλουμε να διαβάσουμε τα συμβάντα και το πεδίο timeMax την τελική. Το API θα μας επιστρέψει δηλαδή μια λίστα με τα συμβάντα μεταξύ των ημερομηνιών timeMin και timeMax.

2.4. Back end

Το κομμάτι που συνδέει τα δύο παραπάνω κομμάτια και κάνει τις υπόλοιπες απαραίτητες ενέργειες είναι το back end. Με τον όρο back end εννοούμε ουσιαστικά το κομμάτι της εφαρμογής που δεν έχει πρόσβαση ο χρήστης. Όπως ανέφερα προηγουμένως το κομμάτι του back end αναπτύχθηκε σε περιβάλλον Django με τη γλώσσα προγραμματισμού Python. Πιο αναλυτικά τα βήματα της διαδικασίας ανάπτυξης της εφαρμογής και screenshots με κομμάτια του κώδικα θα αναφερθούν στο επόμενο κεφάλαιο. Ας εξηγήσουμε πρώτα όμως με λίγα λόγια τις εργασίες που κάνει το back end.

Φόρτωση σελίδων

Αρχικά το back end φορτώνει τις σελίδες που βλέπει ο χρήστης. Το Django χρησιμοποιεί το μοντέλο MTV το οποίο, παρόμοια με το MVC, κάθε φορά που ο χρήστης προσπαθεί να φορτώσει μια σελίδα, καλείται το αντίστοιχο View. Κάποια Views το μόνο που χρειάζεται να κάνουν είναι να φορτώσουν την αντίστοιχη σελίδα. Κάποια κάνουν πιο περίπλοκα πράγματα όπως επεξεργασία δεδομένων, χειρίζονται τις φόρμες που συμπληρώνει

ο χρήστης ή επικοινωνούν με το Google API. Πάμε να δούμε πιο αναλυτικά τις εργασίες της κάθε μεθόδου.

Index: Για να εμφανίσει τη σωστή λίστα με ημερολόγια στο χρήστη, αλλά και για να προσθέσει τα συμβάντα η εφαρμογή πρέπει όπως ανέφερα πιο πάνω να έχει πρόσβαση σε κάποια δεδομένα από το λογαριασμό Google του χρήστη. Αν ο χρήστης έχει ξανά χρησιμοποιήσει την εφαρμογή και ξέρει ότι θέλει σίγουρα να συνδέσει το λογαριασμό του, τότε μπορεί να συνδεθεί και να ξεκινήσει τη διαδικασία. Τι γίνεται όμως αν ο χρήστης χρησιμοποιεί την εφαρμογή για πρώτη φορά; Γι αυτό υπάρχει αυτή η σελίδα. Είναι η πρώτη σελίδα που φορτώνει στο χρήστη και είναι μια σελίδα με απλό κείμενο που περιέχει πληροφορίες για το τι ακριβώς κάνει η κάθε σελίδα της εφαρμογής ώστε να έχει ο χρήστης όλες τις πληροφορίες που χρειάζεται για να αποφασίσει αν θέλει να επιτρέψει στην εφαρμογή πρόσβαση στο λογαριασμό του. Όσο για το τι κάνει το View, σε αυτή την περίπτωση απλώς φορτώνει τη σελίδα.

Log in: Όσον αφορά το back end της εφαρμογής, για τη διαδικασία του Log in δε χρειάζεται να κάνει πολλά. Επειδή ο χρήστης συνδέεται με λογαριασμό της Google, η διαδικασία της ταυτοποίησης γίνεται στέλνοντας το χρήστη στο server της Google όπως εξήγησα παραπάνω στο κομμάτι του OAuth2.

Import csv: Ένα από τα view που κάνει παραπάνω πράγματα από απλά να φορτώνει την ιστοσελίδα στο χρήστη. Χωρίζεται σε δύο μέρη, αναλόγως τι προσπαθεί να κάνει ο χρήστης. Το View καλείται ή με GET ή με POST. Με την GET καλείται όταν ο χρήστης προσπαθεί απλά να φορτώσει τη σελίδα και με POST όταν ο χρήστης προσπαθεί να ανεβάσει το CSV με τα συμβάντα που θέλει να προσθέσει στο Google Calendar. Όταν το View καλείται με GET, αρχικά καλείται μια μέθοδος από το Google API, η calendarList που διαβάζει όλα τα ημερολόγια του χρήστη και μετά τα βάζει σε μια λίστα. Έπειτα φορτώνει τη σελίδα για να τη δει ο χρήστης προσθέτοντας τη λίστα με τα ημερολόγια του χρήστη στη φόρμα για το ανέβασμα του CSV.

Όταν το View καλείται με POST, σημαίνει ότι ο χρήστης έχει ανεβάσει κάποιο αρχείο CSV, έχει διαλέξει σε ποιο ημερολόγιο θέλει να προσθέσει τα συμβάντα και έχει πατήσει το κουμπί upload. Η μέθοδος παίρνει τα δεδομένα αυτά από τη φόρμα, διαβάζει το CSV αρχείο και βάζει τα περιεχόμενα σε μια λίστα και ξεκινάει να τα προσθέτει στο ημερολόγιο που επέλεξε ο χρήστης. Συγκεκριμένα, για το κάθε συμβάν, καλεί μια μέθοδο που φτιάχνει όλες τις πληροφορίες για το συμβάν σε ένα event resource και καλεί το API για να δημιουργήσει το συμβάν και να το προσθέσει στο σωστό ημερολόγιο. Αφού προσθέσει όλα τα συμβάντα φορτώνει ξανά τη σελίδα.

Export csv: Παρόμοια με το Import csv το View χωρίζεται σε δύο μέρη, αναλόγως τι προσπαθεί να κάνει ο χρήστης. Όταν ο χρήστης προσπαθεί απλά να φορτώσει τη σελίδα, καλείται με GET. Εδώ και πάλι καλείται η μέθοδος calendarList, η οποία διαβάζει όλα τα ημερολόγια του χρήστη και μετά τα βάζει σε μια λίστα. Έπειτα φορτώνει τη σελίδα για να τη δει ο χρήστης.

Όταν το View καλείται με POST, σημαίνει ότι ο χρήστης θέλει να κατεβάσει ένα CSV αρχείο με τα συμβάντα ανάμεσα στις δύο ημερομηνίες από το ημερολόγιο που διάλεξε. Η μέθοδος παίρνει τα δεδομένα από τη φόρμα και καλεί το Google Calendar API δίνοντας του τις δύο ημερομηνίες και το ημερολόγιο και το API επιστρέφει μια λίστα με τα συμβάντα. Το view μετά δημιουργεί ένα CSV και αρχίζει μια επανάληψη για να γράψει το κάθε συμβάν στο αρχείο. Μόλις ολοκληρώσει τη διαδικασία ανοίγει ένα παράθυρο στο χρήστη για να επιλέξει που θέλει να αποθηκεύσει το CSV αρχείο. Τέλος φορτώνει ξανά τη σελίδα.

Logout: Με το Django η διαδικασία της αποσύνδεσης του χρήστη είναι αυτοματοποιημένη. Αρκεί να προσθέσουμε την αντίστοιχη βιβλιοθήκη του Django και μετά να φτιάξουμε ένα view που θα περιέχει την εντολή logout(request) και θα καλείται όταν θέλει να αποσυνδεθεί ο χρήστης.

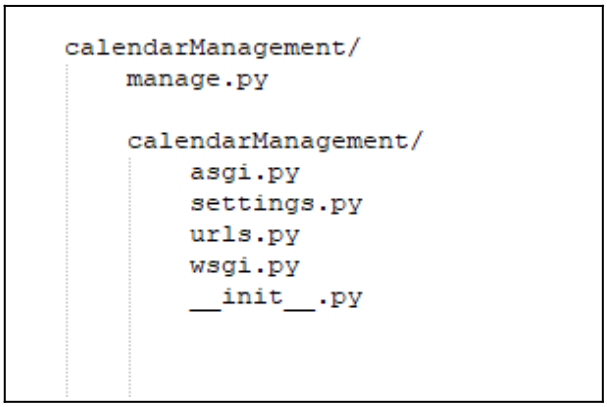
Στο επόμενο κεφάλαιο αναλύεται παραπάνω ο κώδικας της εφαρμογής και αναφέρονται πιο συγκεκριμένα τα views, το Google API και οι βιβλιοθήκες που χρησιμοποιήθηκαν.

3. Εγγραφής της εφαρμογής και παραδείγματα κώδικα

Σε αυτό το κεφάλαιο θα αναλύσω πιο συγκεκριμένα τον κώδικα της εφαρμογής μαζί με εικόνες από κομμάτια του που αξίζει να αναφερθούν. Πριν την ανάλυση αξίζει να αναφέρω ότι η εφαρμογή είναι γραμμένη σε Django 3.0 και Python 3.7.

3.1. Δημιουργία Project και Application

Αρχικά πρέπει να δημιουργήσουμε το project για την εφαρμογή. Το Django δημιουργεί αυτόματα ένα directory με τα αρχεία που θα χρειαστούμε με την εντολή "*django-admin startproject calendarManagement*" την οποία εκτελούμε από το command prompt/terminal με path το φάκελο που θέλουμε να δημιουργηθεί το project. Η εντολή αυτή δημιουργεί τη παρακάτω δομή αρχείων.



Μια μικρή επεξήγηση για το τι κάνει το κάθε αρχείο. Το __init__.py είναι ένα κενό αρχείο που λέει στην Python ότι το συγκεκριμένο directory είναι ένα πακέτο Python. Το settings.py περιέχει όλα τα settings της εφαρμογής. Σε αυτό το αρχείο δηλώνουμε οποιαδήποτε εφαρμογή είναι μέρος του project, τη βάση που χρησιμοποιείται κλπ. Το αρχείο urls.py περιέχει πληροφορίες για το ποιο view καλείται σε κάθε url. Τα wsgi.py και asgi.py βοηθούν την εφαρμογή να επικοινωνεί με το web server (χρησιμοποιείται ένα από τα δύο αναλόγως το server). Το manage.py χρησιμοποιείται για να δημιουργήσει εφαρμογές στο

project, να γίνουν αλλαγές στη βάση δεδομένων και να ξεκινήσει ο web server που παρέχει το Django για τη διάρκεια του development.

Έπειτα, τρέχοντας την επόμενη εντολή στο ίδιο path που βρίσκεται το manage.py δημιουργείται η εφαρμογή manager: "*python3 manage.py startapp manager*". Η εντολή δημιουργεί ένα καινούριο φάκελο και και προσθέτει εκεί κάποια αρχεία που θα χρησιμέψουν για τη συγγραφή του κώδικα της εφαρμογής. Τα περισσότερα αρχεία έχουν ένα όνομα που δείχνει τη χρήση τους (π.χ στο αρχείο views.py θα προστεθούν τα views, στο models.py τα models, κ.λ.π) και περιέχουν ένα μικρό δείγμα κώδικα.

Το δέντρο αρχείων τώρα είναι:

```
calendarManagement/
    manage.py

    calendarManagement/

    manager/
        admin.py
        apps.py
        models.py
        tests.py
        views.py
        __init__.py

        migrations/
            __init__.py
```

Ο φάκελος migrations, χρησιμοποιείται για να αποθηκεύει "migrations" η εφαρμογή. Τα αρχεία migrations επιτρέπουν τα αυτόματα update στη βάση δεδομένων όταν αλλάζουν τα models.

Τώρα που δημιουργήσαμε την εφαρμογή πρέπει να την κάνουμε register στο project ώστε να ξέρει ότι είναι μέρος του. Προσθέτοντας τον κώδικα: *'manager.apps.ManagerConfig'* στο

INSTALLED_APPS του αρχείου settings.py προσδιορίζουμε ένα αντικείμενο που δημιουργήθηκε αυτόματα στο /calendarManagement/manager/apps.py όταν δημιουργήσαμε την εφαρμογή.

3.2. Urls

Το Django δημιουργεί ένα αρχείο URL mapper (urls.py) στο φάκελο του project. Παρόλο που μπορεί να χρησιμοποιηθεί για όλα τα URL mappings, είναι συνήθως καλύτερα η κάθε εφαρμογή να έχει το δικό τις αρχείο urls.py. Στο αρχείο urls.py που βρίσκεται στο/calendarManagement/calendarManagement/urls.py, προσθέτουμε τον παρακάτω κώδικα.

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('manager/', include('manager.urls')),
    path('', include('social_django.urls', namespace='social')),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Εικόνα 1: Αρχείο calendarManagement/calendarManagement/urls.py

Το path admin/ το προσθέτει αυτόματα το Django όταν δημιουργούμε ένα project και είναι για τη σελίδα που βλέπει ο administrator, αν υπάρχει. Στη συγκεκριμένη εργασία δεν υπήρχε λόγος δημιουργίας μιας τέτοιας σελίδας, απλά το άφησα για τυχών μελλοντικές προσθήκες στον κώδικα. Το path manager/ είναι το path για την εφαρμογή manager που δημιουργήσαμε και εκεί που θα βρίσκονται τα URL για τις ιστοσελίδες που θα βλέπει ο χρήστης. Το επόμενο και τελευταίο path είναι μέρος της διαδικασίας Log in με το Google και της βιβλιοθήκης που χρησιμοποίησα. Η τελευταία γραμμή κώδικα είναι για να γνωρίζει το Django από που να φορτώσει τα static αρχεία που χρησιμοποιούνται στην εφαρμογή (εικόνες, CSS, κ.λ.π.).

Φτιάχνουμε ένα urls.py στο φάκελο manager που είναι η εφαρμογή και γράφουμε τον εξής κώδικα.

```
from django.urls import path

from . import views


urlpatterns = [
    path('', views.index, name='index'),
    path('logout/', views.logoutView, name='logout'),
    path('import/', views.importCSV, name='importCSV'),
    path('export/', views.exportCalendar, name='exportCalendar'),
    path('help/', views.helpPage, name='helpPage'),

]
```

Εικόνα 2: Αρχείο /calendarManagement/manager/urls.py

Το συγκεκριμένο αρχείο περιέχει όλα τα URL που χρειάζεται η εφαρμογή για να δει ο χρήστης τις σελίδες που περιέχει. Συγκεκριμένα η πρώτη μεταβλητή είναι το URL που θα βρίσκεται η σελίδα HTML, η δεύτερη είναι το view που θα κληθεί όταν ο χρήστης μεταφερθεί σε αυτό το URL και η τρίτη είναι ένα όνομα που μπορούμε να δώσουμε στο path ώστε να μπορούμε να κάνουμε αναφορά σε αυτό χωρίς να το θυμόμαστε ακριβώς. Για παράδειγμα, όταν ο χρήστης πληκτρολογήσει το URL …com/manager/import/ θα κληθεί το view importCSV.

3.3. Log in και authentication

Για τη διαδικασία του Log in χρησιμοποιήθηκε η βιβλιοθήκη social-auth-app-django. Αφού εγκαταστήσουμε τη βιβλιοθήκη, πρέπει να προσθέσουμε τη γραμμή *'social_django'* στο INSTALLED_APPS στο αρχείο settings.py. Όπως ανέφερα προηγουμένως, το OAuth2 θα χρησιμοποιήσει το server της Google για το authentication οπότε πρέπει να το προσθέσουμε στη λίστα με τα AUTHENTICATION_BACKENDS. Προσθέτουμε λοιπόν τη γραμμή: *'social_core.backends.google.GoogleOAuth2'*. Χρειάζεται επίσης να προσθέσουμε τα LOGIN_URL και LOGIN_REDIRECT_URL γιατί η βιβλιοθήκη τα χρησιμοποιεί για να καθορίσει που θα στείλει

το χρήστη κατά τη διάρκεια και μετά τη διαδικασία του authentication. Τέλος η βιβλιοθήκη χρειάζεται το κλειδί SOCIAL_AUTH_URL_NAMESPACE = ‘social’ στο settings.py.

Για να χρησιμοποιήσουμε το back end της Google για το authentication χρειάζεται να δηλώσουμε ότι η εφαρμογή θα το χρησιμοποιήσει και να πάρουμε κάποια credentials. Συγκεκριμένα, η Google προσφέρει ένα Google Developers Console στο οποίο μπορούμε να δημιουργήσουμε ένα project, να πάρουμε credentials γι’ αυτό (ζευγάρι key και secret) και να δηλώσουμε τα authorized redirect URIs. Τα URI χρησιμοποιούνται από το Google Authentication back end για να κάνουν redirect το χρήστη πίσω στην εφαρμογή μετά το Log in και authentication.

Τέλος ως URL στο κουμπί που θέλουμε να γίνεται το Log in μπορούμε απλώς να βάλουμε την παρακάτω γραμμή..

```
{% else %}
<li class="nav-item">
  <a class="btn btn-primary" href="{% url 'social:begin' 'google-oauth2' %}">Login</a>
</li>
{% endif %}
```

Εικόνα 3: Login URL

3.4. Forms

Μια φόρμα HTML είναι μια ομάδα από πεδία/widgets σε μια ιστοσελίδα, με την οποία μπορούμε να συλλέξουν πληροφορίες από το χρήστη. Είναι ένας σχετικά ασφαλής τρόπος να δεχτεί πληροφορίες ο server, γιατί οι πληροφορίες μπορούν να σταλούν με POST requests.

Το Django απλοποιεί πολύ τη διαδικασία δημιουργίας και διαχείρισης μιας φόρμας προσφέροντας ένα framework που επιτρέπει να ορίζουμε τις φόρμες και τα πεδία τους

προγραμματιστικά και μετά να παράγει η εφαρμογή αυτόματα τον κώδικα HTML. Επίσης δίνει τη δυνατότητα να θέσουμε πως θέλουμε να μετατρέπονται τα δεδομένα της φόρμας αφού τα συλλέξουμε και κάνει αυτόματα τον έλεγχο σε κάθε πεδίο.

```python
from django import forms

class ImportForm(forms.Form):
    upload_csv = forms.FileField(label="CSV file", help_text="", required=True)
    calendar_choices = forms.ChoiceField(choices=[])

    upload_csv.widget.attrs.update({'class': 'form-control-file'})
    calendar_choices.widget.attrs.update({'class': 'form-control'})

    def __init__(self, *args, **kwargs):
        my_arg = kwargs.pop('my_arg')
        super().__init__(*args,**kwargs)
        self.fields['calendar_choices'].choices = my_arg

class ExportForm(forms.Form):
    calendar_choices = forms.ChoiceField(choices=[], required=False, label="")
    timeMin = forms.DateTimeField(input_formats=['%d/%m/%Y %H:%M'], label="From ")
    timeMax = forms.DateTimeField(input_formats=['%d/%m/%Y %H:%M'], label="Until ")

    def __init__(self, *args, **kwargs):
        my_arg = kwargs.pop('my_arg')
        super().__init__(*args,**kwargs)
        self.fields['calendar_choices'].choices = my_arg
```

Εικόνα 4: Περιεχόμενα του αρχείου forms.py

Στην εικόνα φαίνεται ο κώδικας στο αρχείο forms.py το οποίο δημιούργησα και βρίσκεται στο calendarManagement/manager/. Δηλώνοντας την κάθε φόρμα, μπορούμε να προσθέσουμε όλα τα πεδία που θέλουμε να έχει, τα attributes για το κάθε πεδίο αλλά και τις αρχικές τιμές που θα έχουν. Έπειτα για να τις προσθέσουμε στο αρχείο HTML αρκεί απλά να περάσουμε τη σωστή φόρμα μέσω του View που φορτώνει τη σελίδα και να την καλέσουμε από το HTML. Το Django δίνει τη δυνατότητα να καλέσουμε από το HTML συγκεκριμένα πεδία της φόρμας για να μπορούμε να προσθέσουμε tags από το CSS και να καθορίσουμε τη σειρά που θα έχουν τα πεδία.

```
<div class="container">
  <div class="jumbotron">
    <p class="lead">Choose the .csv file you want to upload and the calendar you want to add the events to</p>
    <form name="ImportForm" method="POST" enctype="multipart/form-data">{% csrf_token %}
      <div class="form-row">
        <div class="col">
          <label for="upload_csv">File:</label>
          {{ form.upload_csv }}
        </div>
        <div class="col">
          <label for="calendar_choices">Calendar:</label>
          {{ form.calendar_choices }}
        </div>
      </div>
      <button type="submit" class="btn btn-primary">Upload</button>
    </form>
  </div>
</div>
```

Εικόνα 5: Το HTML της φόρμας για το import

```
<div class="container">
  <div class="jumbotron">
    <p class="lead">Choose between which dates you want to export events</p>
    <form name="ExportForm" method="POST">
      {% csrf_token %}
      <div class="form-row">
        <div class="col">
          <label for="calendar_list">Calendar:</label>
          {{ form.calendar_choices }}
        </div>
        <div class="col">
          <label for="timeMin">From:</label>
          {{ form.timeMin }}
        </div>
        <div class="col">
          <label for="timeMax">Until:</label>
          {{ form.timeMax }}
        </div>
      </div>
        <input type="submit" class="btn btn-primary" value="Export"/>
    </form>
    </div>
  </div>
```

Εικόνα 6: Το HTML της φόρμας για το export

3.5. Templates

Template λέγονται τα αρχεία HTML που δείχνουν στην εφαρμογή πως θα παρουσιάσει τις σελίδες στο χρήστη. Το Django ψάχνει αυτόματα για templates σε έναν ομώνυμο φάκελο στην εφαρμογή. Επειδή ένα μεγάλο κομμάτι της HTML επαναλαμβάνεται σε κάθε σελίδα μπορούμε να χρησιμοποιήσουμε το Django templating language για να δημιουργήσουμε ένα base template και να το κάνουμε extend προσθέτοντας μόνο τα σημεία που είναι διαφορετικά σε κάθε σελίδα.

```
<body>
    <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
      <button class="navbar-toggler" type="button" data-toggle="collapse"
      data-targer="#navbarExample10" aria-controls="navbarsExample10" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggle-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav mr-auto">
          {% block navbar %}{% endblock %}
        </ul>
        <ul class="navbar-nav ml-auto nav-flex-icons">
          {% if user.is_authenticated %}
          <li class="nav-item">
            <div class="mr-2">
              <p class="navbar-text">Welcome, {{ user.username }}</p>
            </div>
          </li>
          <li class="nav-item">
            <a class="btn btn-primary" href="{% url 'logout' %}">Logout</a>
          </li>
          {% else %}
          <li class="nav-item">
            <a class="btn btn-primary" href="{% url 'social:begin' 'google-oauth2' %}">Login</a>
          </li>
          {% endif %}
        </ul>
      </div>
    </nav>

    {% block content %}{% endblock %}

</body>
```

Εικόνα 7: Το body του base_generic.html

Ο κώδικας στην εικόνα περιέχει τα κομμάτια HTML που επαναλαμβάνονται σε κάθε σελίδα με κομμάτια που μπορεί να προστεθεί κώδικας σημειωμένα με τα tags block και endblock. Μπορούμε να προσθέσουμε ένα default content που χρησιμοποιείται όταν δεν επιλέξουμε να βάλουμε κάτι διαφορετικό στις σελίδες που κληρονομούν το base_generic ή να

το αφήσουμε κενό όπως επέλεξα να κάνω εγώ. Όπως βλέπετε το Django επιτρέπει τη χρήστη If στον κώδικα HTML που διευκολύνει και επιταχύνει τη διαδικασία εγγραφής κώδικα.

Αφού έχουμε δημιουργήσει το base_generic.html, θα πρέπει στην αρχή των άλλων html αρχείων να υπάρχει το tag που φαίνεται στην επόμενη εικόνα και να χρησιμοποιηθούν τα block/endblock όπου θέλουμε να αντικατασταθεί ο κώδικας.

```
{% extends "base_generic.html" %}
{% load static %}
```

Εικόνα 8: Tags στην αρχή των αρχείων HTML

Το load static φορτώνει τα στατικά αρχεία ώστε να μπορούν να συμπεριληφθούν στη σελίδα. Μετά από αυτά τα tag, ανάμεσα στα {% block content %}{% endblock %} υπάρχει ο πρόσθετος κώδικας της αντίστοιχης σελίδας και το Django θα ολοκληρώσει την κατασκευή της ιστοσελίδας. Πέρα από των κώδικα για τις φόρμες που είδαμε στο κεφάλαιο 3.4, το υπόλοιπο αρχείο HTML είναι οδηγίες για το χρήστη και κείμενο οπότε δεν αξίζει να αναφερθεί.

Ως CSS χρησιμοποιήθηκε το Bootstrap οπότε αρκεί απλώς να προσθέσουμε το link για το Bootstrap στο head tag του base_generic.html. Όσον αφορά τα στατικά αρχεία, συμπεριλαμβανομένου αρχεία JavaScript, CSS και εικόνες, για να αποφύγουμε τυχών προβλήματα αν αλλάξει η θέση τους, το Django επιτρέπει να προσδιορίσουμε την τοποθεσία τους με μια μεταβλητή που ονομάζεται STATIC_URL και βρίσκεται στο αρχείο settings.py.

3.6. Views και Functions

Τα views είναι μέθοδοι που επεξεργάζονται τα HTTP requests, φορτώνουν τα απαραίτητα δεδομένα από τη βάση δεδομένων, προσθέτουν δεδομένα σε μια HTML σελίδα με βάση ένα template και επιστρέφει το αρχείο HTML που δημιουργήθηκε σε ένα HTTP response για να παρουσιάσει τη σελίδα στο χρήστη. Στο τέλος ενός view καλούμε τη μέθοδο render() για να δημιουργήσουμε μια σελίδα HTML και να την επιστρέψουμε ως response. Η μέθοδος

render() δέχεται τις εξής παραμέτρους: το αρχικό αντικείμενο request, το οποίο είναι ένα HTMLRequest, ένα HTML template και μια παράμετρο context. Η παράμετρος context είναι ένα Python dictionary που περιέχει όσα δεδομένα που θέλουμε να προσθέσουμε στο HTML template.

Τα ποιο απλά views είναι αυτά που φορτώνουν τη σελίδα Help και αυτό που χρησιμοποιεί ο χρήστης για να κάνει αποσύνδεση. Το view για τη σελίδα Help απλώς φορτώνει τη σελίδα στο χρήστη μιας και περιέχει απλό κείμενο και δε χρειάζεται να επεξεργαστεί και να στείλει κάποια δεδομένα. Το view του logout χρησιμοποιεί τη μέθοδο logout που παρέχει το ίδιο το Django οπότε αρκεί απλώς να το καλέσουμε και μετά να φορτώσουμε τη σελίδα που θέλουμε να δει ο χρήστης μετά την αποσύνδεση.

```
def helpPage(request):

    return render(request, 'manager/helpPage.html')

def logoutView(request):
    logout(request)
    return render(request, 'manager/index.html')
```

Εικόνα 9: Views για τη σελίδα Help και τη διαδικασία logout

```
def build_calendar(request):

    user = request.user
    social = user.social_auth.get(provider='google-oauth2')

    creds = google.oauth2.credentials.Credentials(social.extra_data['access_token'])

    GCAL = discovery.build('calendar', 'v3', credentials = creds)
    return GCAL
```

Εικόνα 10: Μέθοδος για τη δημιουργία ημερολογίου

Στη συνέχεια ας δούμε τη μέθοδο που καλούμε όταν θέλουμε να “χτίσουμε” ένα ημερολόγιο. Για να χρησιμοποιήσουμε τις μεθόδους από το Google Calendar API πρέπει πρώτα

να έχουμε ένα ημερολόγιο το οποίο δημιουργούμε με την εντολή discovery.build(όνομα του API, έκδοση του API, credentials του χρήστη). Η μέθοδος build_calendar ζητά τα credentials του χρήστη και μετά φτιάχνει το ημερολόγιο και το επιστρέφει για να μπορούμε να χρησιμοποιήσουμε τις μεθόδους του Google Calendar API.

Ας δούμε το view importCSV. Το view χωρίζεται σε δύο μέρη αναλόγως αν καλείτε με GET ή POST. Όταν καλείται με GET τρέχει ο παρακάτω κώδικας.

```
    GCAL = build_calendar(request)

    calendar_choices.clear()
    while True:

        calendar_list = GCAL.calendarList().list(pageToken=page_token).execute()
        for calendar_list_entry in calendar_list['items']:
            calendar_choices.update({calendar_list_entry['id']: calendar_list_entry['summary']})
        page_token = calendar_list.get('nextPageToken')
        if not page_token:
            break

    form = ImportForm(my_arg=calendar_choices.items())


context = {
    'form': form,
}
return render(request, 'manager/importCSV.html', context)
```

Εικόνα 11: Το κομμάτι του GET στο importCSV view

Συγκεκριμένα, πρώτα καλείται η μέθοδος build_calendar και μετά με το ημερολόγιο που μας επέστρεψε καλούμε τη μέθοδο του API, calendarList() που μας επιτρέπει μια λίστα με τα ημερολόγια του χρήστη. Έπειτα προσθέτουμε τα ημερολόγια στο πεδίο calendar_choices που βρίσκεται στο form της σελίδας importCSV. Τέλος φτιάχνουμε το form και φορτώνουμε τη σελίδα importCSV για το χρήστη.

Όταν καλείται με POST πρώτα τρέχει πάλι η μέθοδος build_calendar. Μετά φορτώνουμε το form και ελέγχουμε αν είναι σωστά συμπληρωμένα τα πεδία. Το Django μπορεί να ελέγξει αυτόματα το form με τη χρήση της μεθόδου is_valid(). Φορτώνουμε το CSV

αρχείο σε μια μεταβλητή και το διαβάζουμε με της χρήση της βιβλιοθήκης του CSV που προσφέρει το Django. Αφού διαβάσουμε και το ημερολόγιο που επέλεξε ο χρήστης τρέχουμε μια επανάληψη και για τη κάθε γραμμή του αρχείου, με τη βοήθεια της μεθόδου populate_calendar που θα δούμε παρακάτω, φτιάχνουμε ένα event resource και το προσθέτουμε στο Google Calendar με το API. Στο τέλος καλείται πάλι η render() που φορτώνει τη σελίδα ξανά στο χρήστη.

```
GCAL = build_calendar(request)
form = ImportForm(request.POST, request.FILES, my_arg=calendar_choices.items())
if form.is_valid():
    fileCSV = request.FILES['upload_csv']

    if fileCSV.name.endswith('.csv'):

        decoded_file = fileCSV.read().decode('utf-8').splitlines()
        reader = csv.DictReader(decoded_file)

        user_calendar = form.cleaned_data['calendar_choices']

        for row in reader:
            event = populate_calendar(row)
            e = GCAL.events().insert(calendarId=user_calendar, sendNotifications=False,
                body=event).execute()
```

Εικόνα 12: Το κομμάτι του POST στο importCSV view

Το view exportCalendar είναι παρόμοιο με το importCSV όσο αναφορά το GET. Πάλι καλείται η μέθοδος build_calendar() και μετά η calendarList() για να γεμίσουμε τη λίστα στη φόρμα της σελίδας με τα ημερολόγια του χρήστη. Μετά φτιάχνουμε το form και φορτώνουμε τη σελίδα.

```
GCAL = build_calendar(request)

calendar_choices.clear()
while True:

    calendar_list = GCAL.calendarList().list(pageToken=page_token).execute()
    for calendar_list_entry in calendar_list['items']:
        calendar_choices.update({calendar_list_entry['id']: calendar_list_entry['summary']})
    page_token = calendar_list.get('nextPageToken')
    if not page_token:
        break

form = ExportForm(my_arg=calendar_choices.items())

context = {
    'form' : form

    }
return render(request, 'manager/exportCalendar.html', context)
```

Εικόνα 13: Κομμάτι GET από το exportCalendar view

```
GCAL = build_calendar(request)

response = HttpResponse(content_type='text/csv')
response['Content-Disposition'] = 'attachment; filename = "calendar.csv"'

form = ExportForm(request.POST, my_arg=calendar_choices.items())
writer = csv.writer(response)

if form.is_valid():
    user_calendar = form.cleaned_data['calendar_choices']
    date_Max = form.cleaned_data['timeMax']
    date_Min = form.cleaned_data['timeMin']

    stringMin = str(date_Min).replace(' ','T')
    stringMax = str(date_Max).replace(' ','T')

    events = GCAL.events().list(calendarId=user_calendar,
        timeMin=stringMin,
        timeMax=stringMax,
        singleEvents=True,
        orderBy='startTime').execute()


    writer.writerow(['event_name', 'date', 'start', 'end', 'attendees', 'location', 'description'])
```

Εικόνα 14: Κομμάτι POST από το exportCalendar view

Όταν καλείται με POST πρέπει να φτιάξουμε ένα αντικείμενο response το οποίο δηλώνει ότι θέλουμε να προσφέρουμε στο χρήστη να αποθηκεύσει ένα CSV αρχείο και να δημιουργήσουμε έναν writer για να μπορούμε να γράψουμε στο CSV. Εδώ και πάλι χρησιμοποιείται η βιβλιοθήκη CSV που προσφέρει το Django. Παίρνουμε τα δεδομένα από τη φόρμα και ελέγχουμε αν είναι σωστά και μετά χρησιμοποιούμε το API για να μας επιστρέψει μια λίστα με τα συμβάντα ανάμεσα στις δύο ημερομηνίες. Γράφουμε τη πρώτη γραμμή του CSV που είναι τα αναγνωριστικά για το τι περιέχει κάθε στήλη και μετά μέσα σε μια επανάληψη γράφουμε το κάθε συμβάν σε μια γραμμή στο CSV. Τέλος με το return response ανοίγει ένα παράθυρο στο χρήστη για να σώσει το αρχείο στον υπολογιστή του.

```
        writer.writerow([event['summary'], date, start[0],
                         end[0], attendees, location, description])


return response
```

Εικόνα 15: Συνέχεια του POST από το exportCalendar view

Η επόμενη μέθοδος λέγετε populate_calendar και παίρνει ως παράμετρο μια λίστα με δεδομένα. Συγκεκριμένα σε αυτή τη μέθοδο στέλνουμε τα δεδομένα από μια γραμμή που έχουμε διαβάσει από το CSV αρχείο που ανέβασε ο χρήστης. Δημιουργούμε ένα EVENT και συμπληρώνουμε τα αντίστοιχα δεδομένα. Τα πεδία summary, start και end είναι υποχρεωτικά και τα location και description είναι απλό κείμενο οπότε ακόμα και να είναι κενά δεν υπάρχει πρόβλημα. Για τα πεδία attendees και recurrence πρέπει να ελέγξουμε αν είναι κενά και να τα επεξεργαστούμε κατάλληλα.

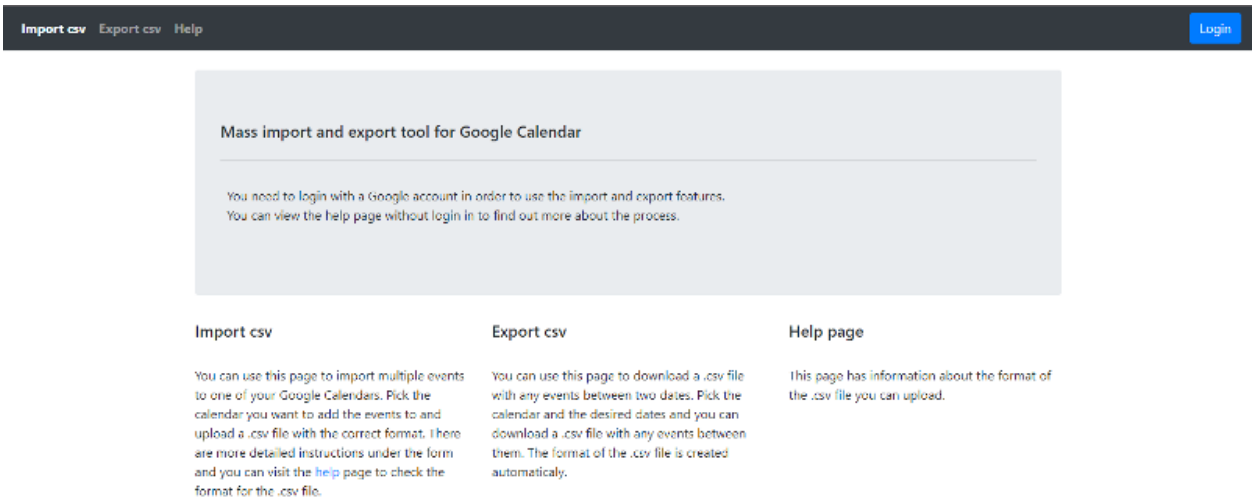
```
def populate_calendar(csv_row):


    EVENT = {
        'summary': csv_row['event_name'],
        'start': {
            'dateTime': csv_row['date']+'T'+csv_row['start']+':00',
            'timeZone' : 'Europe/Athens'
        },
        'end': {
            'dateTime': csv_row['date']+'T'+csv_row['end']+':00',
            'timeZone' : 'Europe/Athens'},

        'location': csv_row['location'],
        'description': csv_row['description'],
    }
    if csv_row['professors'] != '':
        temp = []
        professors = csv_row['professors'].split(',')

        for professor in professors:
            professor = professor.replace(' ','')
            temp.append({'email': professor})
        #'attendees': [{ 'email' : csv_row['professors'] }],
        EVENT['attendees'] = temp

    if csv_row['recurrence'] != '':
        until = csv_row['recurrence'].replace('-','')
        recurrence = ['RRULE:FREQ=WEEKLY;UNTIL='+until+'T230000Z']
        EVENT['recurrence'] = recurrence

    return EVENT
```

Εικόνα 16: Μέθοδος για τη δημιουργία ενός event resource

4. Παράδειγμα εφαρμογής και deployment

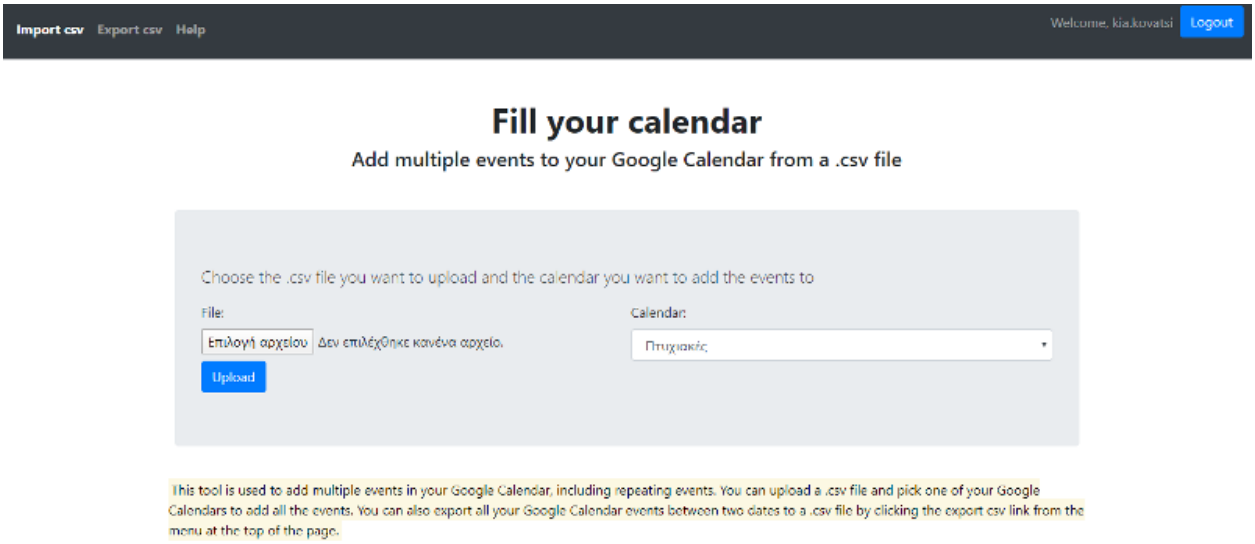
Αφού εξηγήσαμε πως ακριβώς είναι ο κώδικας της εφαρμογής, πάμε να δούμε πως φαίνεται στο χρήστη δίνοντας ένα παράδειγμα χρήσης της και να εξηγήσουμε πως μπορούμε να ανεβάσουμε την εφαρμογή σε έναν public server.

Αρχικά ο χρήστης βρίσκεται σε αυτή τη σελίδα. Η σελίδα εξηγεί με λίγα λόγια τι κάνει η εφαρμογή και δίνει στο χρήστη την ευκαιρία να αποφασίσει αν θέλει να συνδέσει το Google λογαριασμό του για να τη χρησιμοποιήσει.

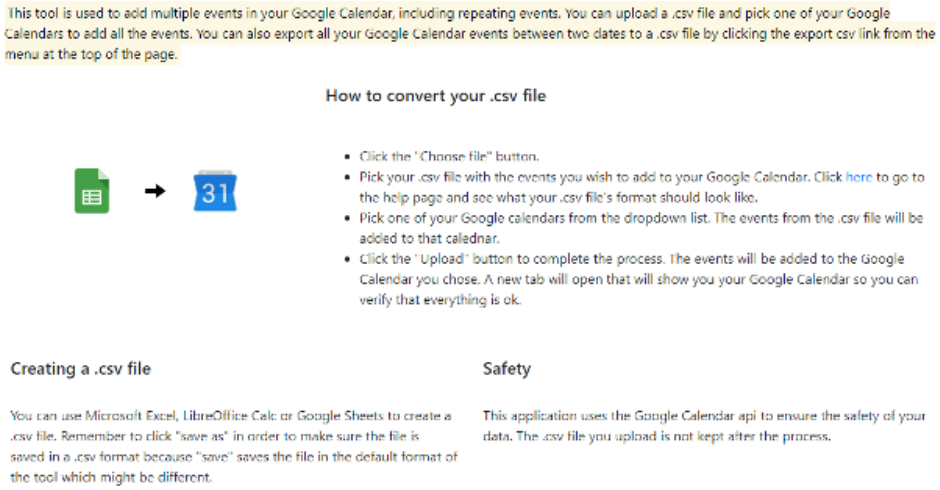


Εικόνα 17: Σελίδα index. Πρώτη σελίδα που έρχεται σε επαφή ο χρήστης

Πατώντας το κουμπί login ο χρήστης μεταφέρεται σε περιβάλλον της Google και αφού ολοκληρώσει τη διαδικασία ξαναγυρίζει πίσω στο site και συγκεκριμένα στην παρακάτω σελίδα.



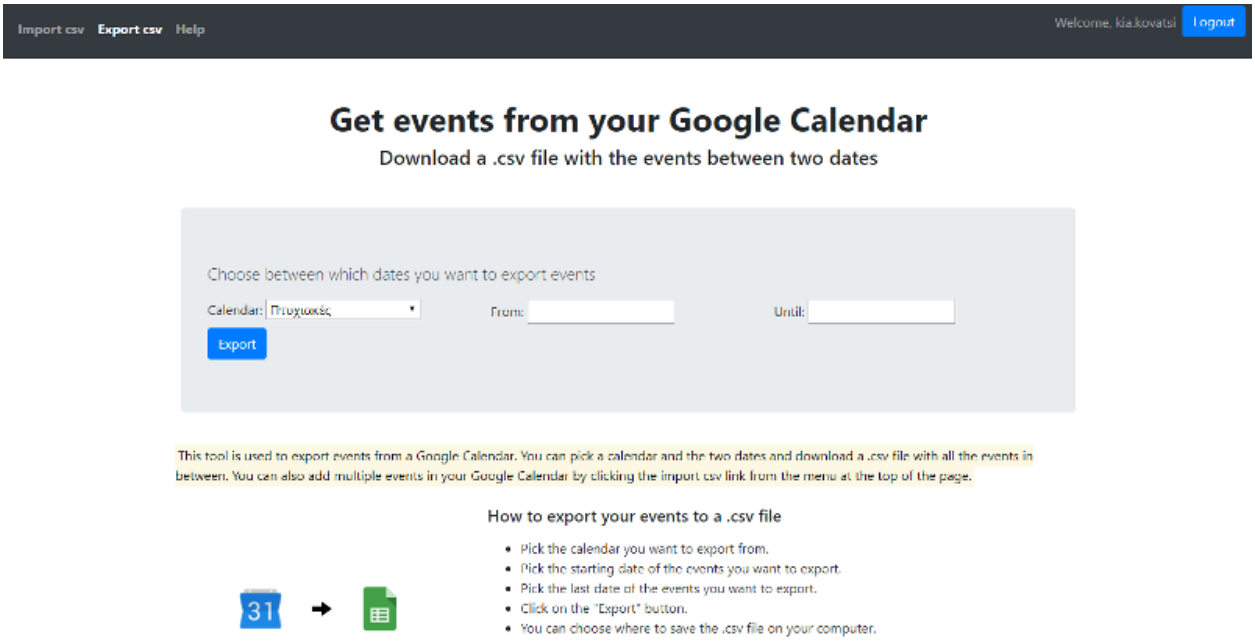
Εικόνα 18: Σελίδα Import csv (μέρος 1)



Εικόνα 19: Σελίδα Import csv (μέρος 2)

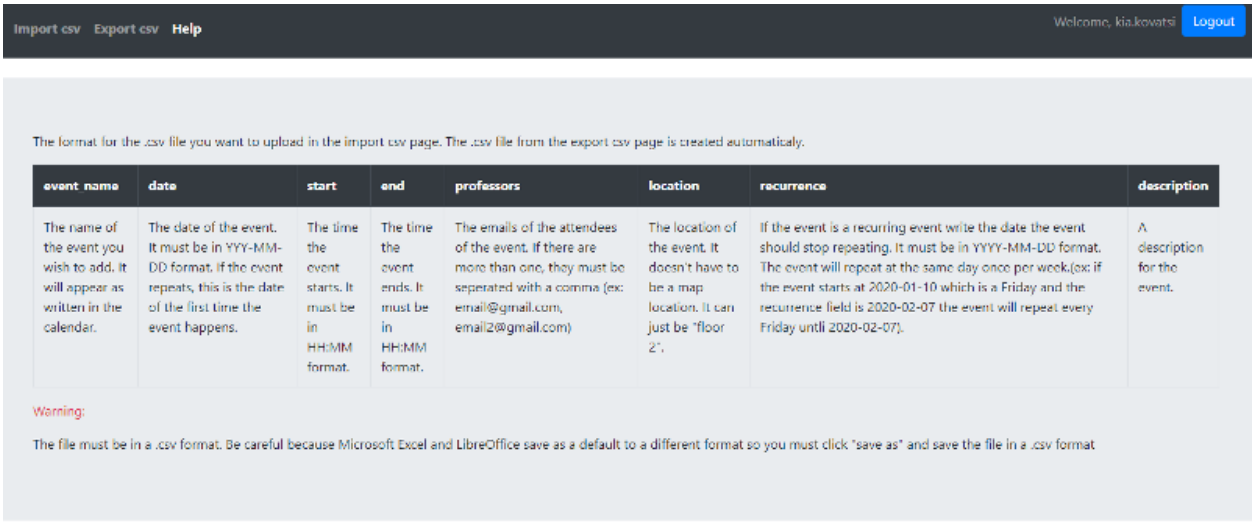
Πατώντας το κουμπί επιλογή αρχείου μπορούμε να διαλέξουμε ένα CSV αρχείο για να ανεβάσουμε. Μετά διαλέγουμε το ημερολόγιο που θέλουμε από τη λίστα και πατάμε το κουμπί upload. Η εφαρμογή ολοκληρώνει τη διαδικασία και φορτώνει ξανά τη σελίδα. Κάτω από τη φόρμα υπάρχουν οδηγίες για την ολοκλήρωση της διαδικασίας.

Αν πατήσουμε το Export csv από το menu στο πάνω μέρος της σελίδας θα βρεθούμε στη σελίδα της επόμενης εικόνας. Αρχικά διαλέγουμε το ημερολόγιο που θέλουμε και μετά τις ημερομηνίες μεταξύ των οποίον θέλουμε να είναι τα συμβάντα και πατάμε το κουμπί export. Ανοίγει ένα παράθυρο για να διαλέξουμε που θέλουμε να αποθηκεύσουμε το αρχείο. Όπως και στην προηγούμενη σελίδα, κάτω από τη φόρμα υπάρχουν οδηγίες προς το χρήστη για να ολοκληρώσει επιτυχώς τη διαδικασία.



Εικόνα 20: Σελίδα Export csv

Τέλος υπάρχει και η σελίδα Help που περιέχει οδηγίες για το πως θα πρέπει να είναι το αρχείο CSV ώστε να μην υπάρξουν λάθη.



Εικόνα 21: Σελίδα Help

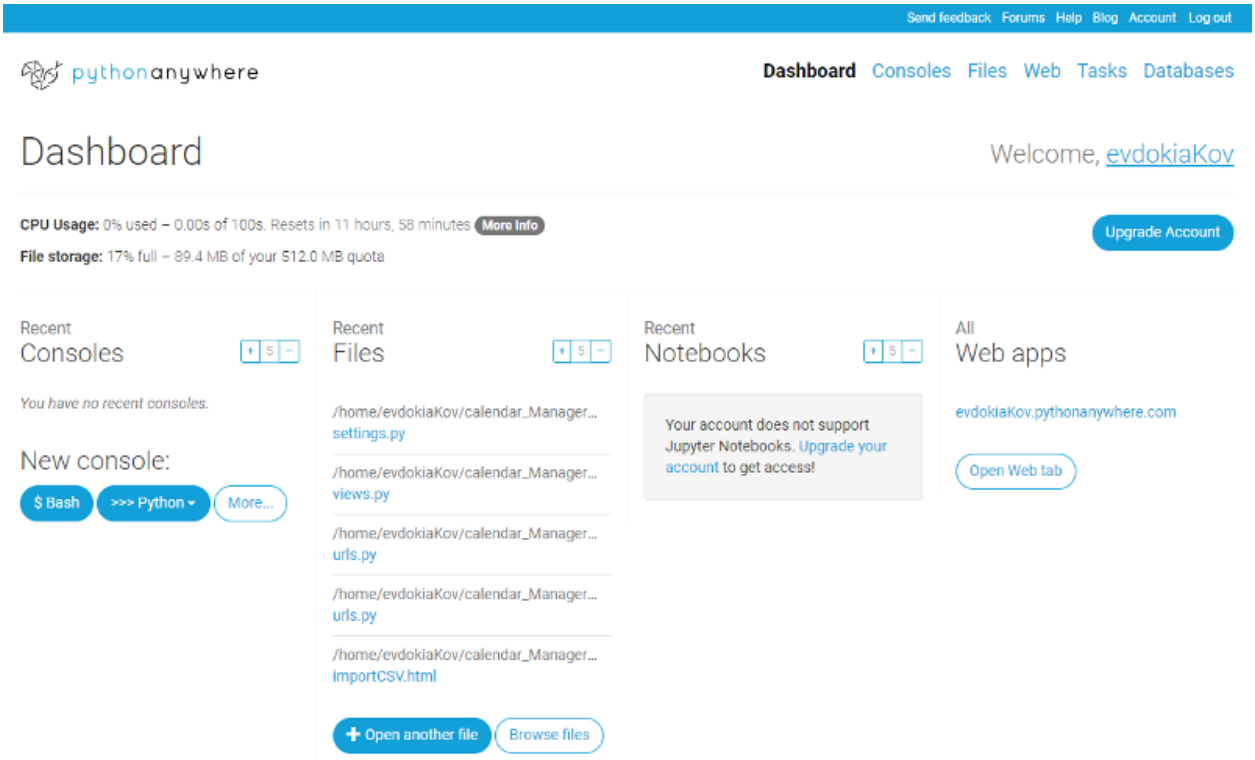
Ο πλήρες κώδικας της εφαρμογής βρίσκεται στο: https://github.com/EvdokiaKovatsi/calendar_Manager

4.1. Deploy

Μετά την υλοποίηση της εφαρμογής και αφού λειτουργεί με επιτυχία σε τοπικό μηχάνημα μέσω του development server που παρέχει το Django, μπορούμε να επιτρέψουμε σε άλλους να τη χρησιμοποιήσουν εγκαθιστώντας τη σε έναν δημόσιο web server. Για να γίνει αυτό πρέπει να αλλάξουμε μερικά πράγματα στο αρχείο settings καθώς και να βρούμε και να κάνουμε set up έναν server που να υποστηρίζει project φτιαγμένα σε περιβάλλον Django. Υπάρχουν πολλοί providers που μπορούμε να διαλέξουμε αναλόγως τις απαιτήσεις που έχουμε αλλά για τη συγκεκριμένη περίπτωση μας αρκεί ένας που παρέχει κάποιες βασικές λειτουργίες δωρεάν μιας και το χρειαζόμαστε απλώς για να εξοικειωθούμε με τη διαδικασία και να δοκιμάσουμε την εφαρμογή. Ο server που επιλέχθηκε είναι το Python Anywhere.

Αφού δημιουργήσουμε ένα account στο Python Anywhere, μπορούμε από το dashboard που έχει να ανοίξουμε μια κονσόλα bash που παρέχεται από το Python Anywhere

για να εγκαταστήσουμε τα πακέτα που χρησιμοποιεί η εφαρμογή. Για να μην υπάρξουν προβλήματα σε περίπτωση που το Python Anywhere κάνει αυτόματα update σε πακέτα καλό είναι να δημιουργήσουμε ένα virtual environment στο οποίο θα είναι εγκατεστημένα τα πακέτα που χρησιμοποιούμε και εκεί θα τρέχει η εφαρμογή μας.



Εικόνα 22: Dashboard του Python Anywhere

Έπειτα από το αντίστοιχο tab στο πάνω μέρος της σελίδας δημιουργούμε ένα νέο Web app, στο οποίο προσθέτουμε το path για το virtual environment που δημιουργήσαμε. Το Python Anywhere θα δημιουργήσει αυτόματα ένα domain για την εφαρμογή μας το οποίο επειδή το χρησιμοποιούμε δωρεάν είναι απλά username.pythonanywhere.com. Μπορούμε τώρα να ανεβάσουμε τα αρχεία της εφαρμογής εύκολα στο server από το GitHub με την εντολή git clone.

Αφού προσθέσουμε το project στο server πρέπει να κάνουμε μερικές μετατροπές στο αρχείο settings.py της εφαρμογής. Ευτυχώς το ίδιο το Django προσφέρει στο documentation

οδηγίες για ότι χρειάζεται να αλλάξει πριν κάνουμε deploy. Συγκεκριμένα, πρέπει να αλλάξουμε τη μεταβλητή Debug από True σε False και να προσθέσουμε στη λίστα ALLOWED_HOSTS το domain που μας παρέχει ο server. Τέλος προσθέτουμε τις γραμμές: CSRF_COOKIE_SECURE = True, SESSION_COOKIE_SECURE = True και δηλώνουμε στη μεταβλητή STATIC_ROOT το path που θα είναι όλα τα static αρχεία. Για να είμαστε σίγουροι ότι όλα τα αρχεία static είναι στο σωστό φάκελο, τρέχουμε από την κονσόλα του Python Anywhere την εντολή *python manage.py collectstatic*. Η εντολή αυτή θα συλλέξει όλα τα αρχεία static από διάφορα σημεία του project στο φάκελο που έχουμε δηλώσει στο STATIC_ROOT στο αρχείο settings.py. Επίσης προσθέτουμε το path του φακέλου static στο app που δημιουργήσαμε στην καρτέλα web που βρίσκονται όλες οι πληροφορίες για την εφαρμογή.

Τέλος πρέπει να αλλάξουμε το αρχείο wsgi.py που υπάρχει στην καρτέλα Web. Ανοίγοντας το αρχείο βλέπουμε ότι υπάρχει ήδη ο κώδικας που χρειαζόμαστε απλά είναι σε σχόλια. Βγάζουμε τον κώδικα από τα σχόλια και προσθέτουμε το path για το project και για το αρχείο settings.py. Στην καρτέλα Web φορτώνουμε το site και μπορούμε να χρησιμοποιήσουμε την εφαρμογή.

5. Συμπεράσματα

Ανακεφαλαιώνοντας, η ανάγκη της συγκεκριμένης εφαρμογής δημιουργήθηκε από την επιθυμία εισαγωγής πολλαπλών συμβάντων σε ένα ημερολόγιο Google. Για τους χρήστες που έχουν ανάγκη να δημιουργήσουν πολλά συμβάντα ταυτόχρονα και συγκεκριμένα επαναλαμβανόμενα δεν υπάρχει κάποια ήδη υπάρχουσα εφαρμογή που να τους λύνει το συγκεκριμένο πρόβλημα. Αυτό ξεκινήσαμε να λύσουμε, προσθέτοντας επίσης την επιλογή να δημιουργήσει ο χρήστης ένα CSV αρχείο με τα συμβάντα μεταξύ δύο ημερομηνιών που βρίσκονται σε ένα από τα ημερολόγια του.

Η εφαρμογή δημιουργήθηκε σε περιβάλλον Django με τη χρήση της Python. Το Django προσφέρει πολλά features για τη διευκόλυνση και την ταχύτερη διαδικασία του development με ένα από τα σημαντικότερα να είναι ένας έτοιμος local server για τη διαδικασία testing του κώδικα. Για την επικοινωνία με το Google Calendar έγινε χρήση του Google Calendar API και του OAuth2. Αυτά μας έδωσαν τη δυνατότητα να έχουμε πρόσβαση στα Google ημερολόγια του χρήστη με ασφαλή τρόπο χωρίς να βλέπουμε προσωπικά δεδομένα που δεν επιθυμεί ο χρήστης να μοιραστεί. Καταφέραμε τελικά να φτιάξουμε μια εφαρμογή που μπορεί να ολοκληρώσει τις εργασίες που είχαμε θέσει ως στόχους. Ο χρήστης μπορεί να ανεβάσει ένα αρχείο CSV με τα συμβάντα που θέλει να δημιουργήσει και η εφαρμογή τα προσθέτει στο Google Calendar που επέλεξε. Μπορεί επίσης να διαλέξει δύο ημερομηνίες και να δημιουργήσει ένα αρχείο CSV με τα συμβάντα αποθηκευμένα στο επιλεγμένο Google Calendar μεταξύ αυτών των ημερομηνιών. Τέλος είδαμε πως μπορούμε να ανεβάσουμε το project μας σε έναν δημόσιο server ώστε η εφαρμογή μας να είναι προσβάσιμη στο ευρύ κοινό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. MDN (2019, November 30-a) Django Introduction. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

2. MDN (2019, December 8-b) Django Tutorial Part 9: Working with forms. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Forms

3. wescpy. (2015, September 9). Creating events in Google Calendar from Python. [Blog Post]. Retrieved from http://wescpy.blogspot.com/2015/09/creating-events-in-google-calendar.html

4. Lamas M. (2018, May 28). OAuth Authentication in Django with social-auth. Retrieved from https://medium.com/trabe/oauth-authentication-in-django-with-social-auth-c67a002479c1

5. Reinke, J. (2016, January 22). Understanding OAuth2. Retrieved from http://www.bubblecode.net/en/2016/01/22/understanding-oauth2/

6. Galitz, W. O (2007). *The Essential Guide to User Interface Design. An Introduction to GUI Design Principles and Techniques* (3rd ed.). Indianapolis, IN: Wiley Publishing.

7. Strizver I. (2014) *Type rules!: the designer's guide to professional typography* (4th ed.). (pp. 37-158). Hoboken, NJ: John Wiley & Sons.

Για τη συγγραφή κώδικα

Tutorials για το Django και το Google Calendar API:

1. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website
2. https://tutorial.djangogirls.org/en/django_start_project/
3. http://wescpy.blogspot.com/2015/09/creating-events-in-google-calendar.html
4. https://medium.com/trabe/oauth-authentication-in-django-with-social-auth-c67a002479c1

Official documentation:

1. https://developers.google.com/calendar/v3/reference/events?hl=en
2. https://developers.google.com/calendar/v3/reference/calendarList?hl=en
3. https://docs.djangoproject.com/en/3.0/intro/tutorial01/
4. https://docs.djangoproject.com/en/3.0/topics/auth/default/
5. https://docs.djangoproject.com/en/3.0/howto/outputting-csv/
6. https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/
7. https://getbootstrap.com/docs/4.3/layout/grid/

Από το stackoverflow.com:

1. https://stackoverflow.com/questions/49231442/google-calendar-api-how-to-find-attendee-list
2. https://stackoverflow.com/questions/51771235/google-drive-access-from-django-python/51911064
3. https://stackoverflow.com/questions/37704491/integrating-google-sign-in-and-calendar-access-in-an-django-application