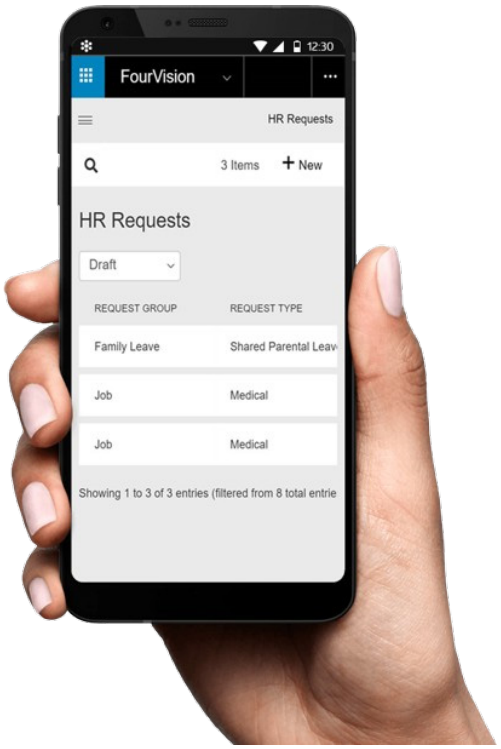


ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΥΠΟΒΟΛΗ ΑΙΤΗΣΕΩΝ ΤΩΝ ΑΔΕΙΩΝ

Web Application Development

ΕΦΠ02: **Υπολογιστικές και Δικτυακές Υποδομές Ι**

George Krasakis
AM: Ap23012
Διδάσκων: Ανάργυρος Τσαδήμας

Περιεχόμενα

1. Εισαγωγή

1.1 Επισκόπηση

Η διαδικτυακή εφαρμογή που βασίζεται στο Flask στοχεύει στην απλοποίηση της διαχείρισης αιτήσεων αδειών για οργανισμούς. Παρέχει μια εύχρηστη διεπαφή για τους υπαλλήλους και τους διαχειριστές για την αποτελεσματική υποβολή, παρακολούθηση και διαχείριση των αιτήσεων άδειας. Ένα από τα βασικά χαρακτηριστικά της εφαρμογής είναι η ικανότητά της να χειρίζεται απρόσκοπτα τα αιτήματα αδειών. Οι εργαζόμενοι μπορούν εύκολα να υποβάλλουν τα αιτήματα αδείας τους μέσω μιας φιλικής προς το χρήστη φόρμας, παρέχοντας βασικές λεπτομέρειες, όπως η ημερομηνία έναρξης, η ημερομηνία λήξης και ο λόγος της άδειας. Οι διαχειριστές, από την άλλη πλευρά, μπορούν να εξετάζουν και να διαχειρίζονται αποτελεσματικά αυτά τα αιτήματα, διασφαλίζοντας την ομαλή λειτουργία του οργανισμού.

Για τη βελτίωση της εμπειρίας του χρήστη και της οπτικής ελκυστικότητας της εφαρμογής, χρησιμοποιούνται επεκτάσεις Flask όπως Flask-WTF και Flask-Bootstrap. Το Flask-WTF απλοποιεί το χειρισμό φορμών παρέχοντας εύχρηστα εργαλεία για τη δημιουργία, την επικύρωση και την επεξεργασία φορμών. Παράλληλα, το Flask-Bootstrap προσφέρει προσχεδιασμένα στοιχεία και διατάξεις UI, επιτρέποντας τη δημιουργία αισθητικά ευχάριστων διεπαφών χωρίς την ανάγκη εκτεταμένης μορφοποίησης CSS.

2. Εγκατάσταση

2.1 Ρύθμιση Flask και βιβλιοθηκών

Τα θεμέλια της διαδικτυακής εφαρμογής βασίζονται στο λογισμικό Flask, ο οποίο παρέχει τη βασική λειτουργικότητα που απαιτείται για την ανάπτυξη εφαρμογών ιστού. Παράλληλα με το Flask, χρησιμοποιούνται διάφορες βιβλιοθήκες και επεκτάσεις που έχουν ως σκοπό να ενισχύσουν τις δυνατότητες της εφαρμογής και να προσφέρουν έναν πιο απλοποιημένο τρόπο υλοποίησης. Παρακάτω αναλύονται κάποια στοιχεία για τις βιβλιοθήκες που χρησιμοποιήθηκαν στη παρούσα εργασία.

Flask Framework

Σκοπός: Παρέχει τα βασικά εργαλεία που χρειάζονται για το χειρισμό αιτημάτων HTTP, τη δρομολόγηση διευθύνσεων URL και την απόδοση προτύπων.

Χαρακτηριστικά: Το Flask παρέχει ένα απλοποιημένο περιβάλλον για την ανάπτυξη ιστοσελίδων, επιτρέποντας την τμηματική και ευέλικτη σχεδίαση μιας εφαρμογής. Επιπρόσθετα, υποστηρίζει τη δημιουργία RESTful APIs, την κληρονομικότητα προτύπων και τη δρομολόγηση URL μέσω διακοσμητών.

- **RESTful APIs:** Πρόκειται για ένα σύνολο κατευθυντήριων γραμμών για το σχεδιασμό υπηρεσιών ιστού που ακολουθούν τις αρχές (Representational State Transfer - REST).
- **Κληρονομικότητα προτύπων:** Το Flask υποστηρίζει την κληρονομικότητα προτύπων, ένα χαρακτηριστικό που επιτρέπει στο χρήστη-προγραμματιστή να ορίζει ένα βασικό πρότυπο HTML με κοινά στοιχεία και να επεκτείνει ή να παρακάμψει συγκεκριμένα μπλοκ.
- **Δρομολόγηση URL μέσω διακοσμητών:** Για τον ορισμό διαδρομών URL και την αντιστοίχισή τους σε συγκεκριμένες λειτουργίες προβολής στην εφαρμογή. Χρησιμοποιώντας μια συνάρτηση (Decorator) με @app.route('/path'), μπορούν να οριστούν τα μονοπάτια και να χειριστούν αιτήσεις για διαφορετικά URL μέσα στην εφαρμογή.

Flask-WTF

Σκοπός: Εξυπηρετεί τον απλοποιημένο χειρισμό φορμών εντός της εφαρμογής, παρέχοντας τα κατάλληλα εργαλεία για τη δημιουργία και την επικύρωση web forms.

Χαρακτηριστικά: ενσωματώνεται με το Flask, παρέχοντας επικύρωση φορμών, προστασία CSRF και δυνατότητες απόδοσης φορμών. Επιπρόσθετα, βελτιώνει τη δημιουργία τους προσφέροντας τύπους πεδίων, κανόνες επικύρωσης και μηνύματα σφάλματος.

Flask-Bootstrap

Σκοπός: Προσφέρει προ-σχεδιασμένα στοιχεία διεπαφής και διατάξεις UI.

Χαρακτηριστικά: Παρέχονται έτοιμα πρότυπα προς χρήση, φόρμες, κουμπιά και μπάρες πλοήγησης για να δημιουργήσουν διεπαφές χρήστη. Επιπρόσθετα, μειώνει την ανάγκη για χειροκίνητη διαμόρφωση CSS.

Logging Configuration:

Σκοπός: Η καταγραφή πληροφοριών εντοπισμού σφαλμάτων, προειδοποιήσεων που παράγονται από την εφαρμογή κατά τη διάρκεια της εκτέλεσης.

Χαρακτηριστικά: Η ενότητα καταγραφής στην Python επιτρέπει την προσαρμογή της εξόδου καταγραφής, συμπεριλαμβανομένων των επιπέδων καταγραφής, της μορφοποίησης και του προορισμού.

CSRFProtect

Σκοπός: Εφαρμόζονται μέτρα ασφαλείας, όπως η προστασία CSRF και η δημιουργία μυστικού κλειδιού, για την προστασία της εφαρμογής από κοινές ευπάθειες του διαδικτύου.

Χαρακτηριστικά: Το CSRFProtect από το Flask-WTF βοηθά στην αποτροπή επιθέσεων πλαστογράφησης αιτήσεων διασταυρούμενης τοποθεσίας (CSRF), δημιουργώντας και επικυρώνοντας διακριτικά CSRF για κάθε υποβολή φόρμας. Επιπλέον, ένα τυχαία παραγόμενο μυστικό κλειδί χρησιμοποιείται για την ασφάλεια των δεδομένων συνόδου και των κρυπτογραφικών λειτουργιών, ενισχύοντας την ασφάλεια της εφαρμογής.

2.2 Αρχικοποίηση της βάσης δεδομένων

Η αρχικοποίηση της βάσης δεδομένων περιλαμβάνει τη ρύθμιση του σχεδιαγράμματος της βάσης δεδομένων και την αρχικοποίησή της με τυχόν απαιτούμενα δεδομένα ή προεπιλεγμένες τιμές και περιλαμβάνει τα ακόλουθα βήματα:

- Ορισμός σχεδιαγράμματος βάσης

Ορισμός της δομής των πινάκων/οντοτήτων της βάσης δεδομένων και των σχέσεων τους. Αυτό γίνεται με τη χρήση ενός εργαλείου (ORM) όπως το SQLAlchemy, το οποίο επιτρέπει να ορίζουν μοντέλα βάσεων δεδομένων χρησιμοποιώντας κλάσεις Python.

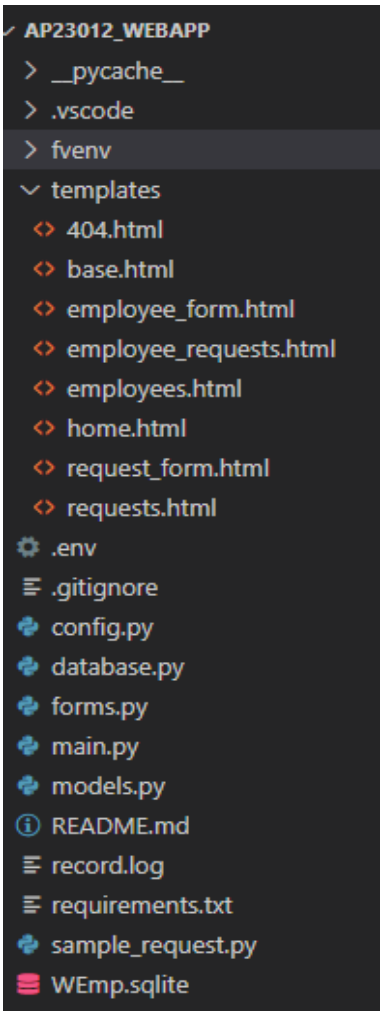
- Δημιουργία βάσης δεδομένων
- Συμπλήρωση αρχικών δεδομένων
- Δημιουργία σύνδεσης
- Διαδικασία αρχικοποίησης

Εκτέλεση της διαδικασίας αρχικοποίησης κατά την εκκίνηση της εφαρμογής Flask. Αυτό μπορεί να γίνει με την καταχώριση συναρτήσεων αρχικοποίησης ή άγκιστρων που εκτελούνται πριν η εφαρμογή αρχίσει να εξυπηρετεί αιτήσεις.

- Χειρισμός σφαλμάτων

3. Δομή εργασίας

Η εργασία που υλοποίησα αποτελείται από τους εξής καταλόγους και τα αρχεία τους:



pycache: Περιέχει αρχεία bytecode Python και χρησιμοποιείται για την επιτάχυνση της φόρτωσης των ενοτήτων.

.vscode: Περιέχει ρυθμίσεις που αφορούν το VS Code. Μπορεί να περιλαμβάνει αρχεία ρυθμίσεων για εργασίες, εντοπισμό σφαλμάτων και άλλες ρυθμίσεις του επεξεργαστή.

fvenv: Φιλοξενεί ένα εικονικό περιβάλλον Python, το οποίο λειτουργεί ως ένα αυτόνομο περιβάλλον ξεχωριστό από την εγκατάσταση Python σε όλο το σύστημα. Σε αυτό το περιβάλλον αποθηκεύονται όλες οι εξαρτήσεις και τα πακέτα του project, διασφαλίζοντας ότι δεν έρχονται σε σύγκρουση με αυτές άλλων project ή του συνολικού περιβάλλοντος Python.

templates: αποθηκεύει πρότυπα HTML που χρησιμοποιούνται από το Flask για τη δυναμική απόδοση ιστοσελίδων. Τα templates μπορεί να περιέχουν placeholders και λογική για τη δημιουργία δυναμικού περιεχομένου με βάση δεδομένα από το διακομιστή.

- **.env**: περιέχει συνήθως μεταβλητές περιβάλλοντος που χρησιμοποιούνται για τη διαμόρφωση της εφαρμογής Flask. Μπορεί να περιλαμβάνει ευαίσθητες πληροφορίες όπως κλειδιά API ή διαπιστευτήρια βάσης δεδομένων.
- **.gitignore**: καθορίζει πρότυπα για αρχεία και καταλόγους που το Git θα πρέπει να αγνοεί κατά την παρακολούθηση των αλλαγών. Χρησιμοποιείται για την εξαίρεση αρχείων όπως προσωρινά αρχεία, αρχεία καταγραφής και καταλόγους που περιέχουν εξαρτήσεις από τον έλεγχο εκδόσεων.

- **config.py**: περιέχει ρυθμίσεις παραμέτρων για την εφαρμογή Flask. Μπορεί να περιλαμβάνει ρυθμίσεις που σχετίζονται με τη σύνδεση με τη βάση δεδομένων, την καταγραφή και την ασφάλεια.
- **database.py**: ορίζει συναρτήσεις και κλάσεις που σχετίζονται με την αρχικοποίηση της βάσης δεδομένων, τη σύνδεση της βάσης δεδομένων και τα μοντέλα της βάσης δεδομένων που χρησιμοποιούν ένα εργαλείο (ORM) όπως το SQLAlchemy.
- **forms.py**: περιέχει κλάσεις που αντιπροσωπεύουν φόρμες ιστού που χρησιμοποιούνται στην εφαρμογή Flask.
- **main.py**: χρησιμεύει ως το κύριο σημείο εισόδου για την εφαρμογή Flask. Καθορίζει τις διαδρομές, τις προβολές και τη λογική της εφαρμογής, συμπεριλαμβανομένου του χειρισμού αιτήσεων, της δημιουργίας απαντήσεων και του χειρισμού σφαλμάτων.
- **models.py**: περιέχει κλάσεις που αντιπροσωπεύουν μοντέλα βάσεων δεδομένων ή οντότητες που χρησιμοποιούνται στην εφαρμογή Flask. Καθορίζει πίνακες βάσης δεδομένων, στήλες, σχέσεις και μεθόδους για την αλληλεπίδραση με τη βάση δεδομένων χρησιμοποιώντας SQLAlchemy ή παρόμοιο ORM.

  sample_request.py: Αυτό το αρχείο Python μπορεί να περιέχει δείγματα κώδικα ή σενάρια για την πραγματοποίηση αιτημάτων HTTP για τη δοκιμή των τελικών σημείων ή της λειτουργικότητας του API της εφαρμογής Flask.
- **Wemp.sqlite**: Αυτό το αρχείο είναι ένα αρχείο βάσης δεδομένων SQLite που χρησιμοποιείται από την εφαρμογή Flask για τη μόνιμη αποθήκευση δεδομένων. Περιέχει πίνακες και δεδομένα που ορίζονται από τα μοντέλα της εφαρμογής και μπορεί να χρησιμοποιείται για την αποθήκευση πληροφοριών χρήστη, αιτήσεων άδειας ή άλλων σχετικών δεδομένων.

3.1 Κώδικας στο main.py

```python
import os

from flask import Flask, redirect, url_for, request, render_template, flash, abort
from flask_bootstrap import Bootstrap5
import logging
from flask_wtf import CSRFProtect
from database import init_db, db_session

from models import Employee, LeaveRequest
from forms import EmployeeForm, LeaveRequestForm


#Configuring Logging to Record Debug Information to 'record.log' File
logging.basicConfig(filename="record.log", level=logging.DEBUG,datefmt= "%Y-%m-%d %H:%M:%S")

#Flask extensions BOOTSTRAP and CSRFProtect
app = Flask(__name__)
app.config.from_object('config')
# Bootstrap-Flask requires this line
bootstrap = Bootstrap5(app)
# Flask-WTF requires this line
csrf = CSRFProtect(app)


#Intialization of the database within the context of a Flask application, ensuring proper con
with app.app_context():
    init_db()


#Setting up a Random Secret Key for Flask Application Security (seasurf token)
#************************************
import secrets

foo = secrets.token_urlsafe(16)
app.secret_key = foo
#************************************

#Database Session Cleanup Function
@app.teardown_appcontext
def shutdown_session(exception=None):
    db_session.close()

#Flask route decorator function
#Define routes and views
```

Το αρχικό κομμάτι του κώδικα στο αρχείο main.py είναι υπεύθυνο για την αρχικοποίηση της εφαρμογής Flask, καθώς ρυθμίζει και διάφορες επεκτάσεις αυτής. Αρχικά καθορίζει το logging για την καταγραφή πληροφοριών και εντοπισμό σφαλμάτων σε ένα αρχείο με όνομα «record.log». Επιπρόσθετα, οι επεκτάσεις Flask Bootstrap5 και CSRFProtect αρχικοποιούνται το μεν για το παρουσιαστικό το άλλο για την προστασία των δεδομένων εισαγωγής σε μια φόρμα. Ακόμη αρχικοποιείται η βάση

δεδομένων και δημιουργείται ένα τυχαίο μυστικό κλειδί για τις ανάγκες ασφάλειας της εφαρμογής. Τέλος, συναντάται μια συνάρτηση που διαχειρίζεται ένα session και διασφαλίζει ότι οι συνδέσεις βάσης δεδομένων αποδεσμεύονται όταν δεν χρειάζονται πλέον, βελτιώνοντας την απόδοση και την αξιοπιστία της εφαρμογής.

```
#(HOME PAGE)
#**************************************************
@app.route('/')
def home():
    return render_template ('home.html')
#**************************************************
```

Όταν ένας χρήστης πλοηγείται στη / διεύθυνση URL της εφαρμογής Flask (π.χ. http://localhost:5000/), εκτελείται η συνάρτηση **home** και το περιεχόμενο του προτύπου home.html αναπαράγεται και εμφανίζεται στο πρόγραμμα περιήγησης ιστού του χρήστη. Το κομμάτι του κώδικα που είναι υπεύθυνο παρουσιάζεται από πάνω.

```
#Employees
#************************************************************************
@app.route("/employees", methods=["GET"])
def show_employees():
    employees = Employee.query.all()
    return render_template("employees.html", employees=employees)
```

Το παραπάνω κομμάτι του κώδικα, αντιστοιχεί σε ένα τελικό /employees το οποίο είναι υπεύθυνο για τον χειρισμό **HTTP GET** αιτήσεων. Όταν γίνεται μια αίτηση **GET** σε αυτό το τελικό σημείο, ανακτά όλους τους υπαλλήλους από τη βάση δεδομένων χρησιμοποιώντας τη μέθοδο **Employee.query.all**(), η οποία επιστρέφει μια λίστα με όλα τα αντικείμενα υπαλλήλων. Αυτά τα αντικείμενα υπαλλήλων περνούν στη συνέχεια στη συνάρτηση **render_template** μαζί με το όνομα του αρχείου προτύπου **HTML employees.html**. Τέλος, η συνάρτηση **render_template** αποδίδει το πρότυπο **HTML employees.html**, περνώντας τη λίστα των υπαλλήλων στο πρότυπο για εμφάνιση.

```python
@app.route("/employee", methods=["GET", "POST"])
def show_employee_form():
    form = EmployeeForm()
    message = ""
    if form.validate_on_submit():
        name = form.name.data
        surname = form.surname.data
        birth_year = form.birth_year.data
        email = form.email.data
        phone = form.phone.data
        job_position = form.job_position.data

        employee = Employee(
            name=name,
            surname=surname,
            birth_year=birth_year,
            email=email,
            phone=phone,
            job_position=job_position
            )
        db_session.add(employee)
        db_session.commit()
        return redirect(url_for("show_employee_form"))
        #return redirect(url_for("show_users", users=users))
    return render_template("employee_form.html", form=form, message=message)
```

Στη περίπτωση αυτή ορίζεται το **/employee** που χειρίζεται **HTTP GET** και **HTTP POST** αιτήσεις. Κατά την πραγματοποίηση μιας αίτησης GET, απεικονίζεται μια φόρμα HTML (**employee_form.html**) με σκοπό να εισάγουν οι χρήστες πληροφορίες για τους υπαλλήλους. Αντίστοιχα σε ένα αίτημα **POST** το οποίο συνδυάζεται με την υποβολή της φόρμας, πρώτα αρχικοποιεί το **EmployeeForm** για να χειριστεί τα δεδομένα της φόρμας. Εάν τα δεδομένα της φόρμας περάσουν την επικύρωση θα πρέπει η **form.validate_on_submit()** να επιστρέψει **True** και στη συνέχεια θα γίνει εξαγωγή των στοιχείων του υπαλλήλου από τα πεδία της φόρμας.

Σε επόμενο στάδιο, δημιουργείται το **Employee** χρησιμοποιώντας τα στοιχεία που εξήχθησαν προηγουμένως και ακολουθεί η προσθήκη του στη βάση δεδομένων **db_session(employee)**. Κατόπιν, της προσθήκης δεσμεύονται οι αλλαγές στη βάση **db_session.commit().**

Τέλος, γίνεται ανακατεύθυνση του χρήστη πίσω στο σημείο /employee χρησιμοποιώντας **redirect(url_for("show_employee_form"))**, το οποίο ενεργοποιεί μια νέα αίτηση GET, καθαρίζοντας ουσιαστικά τη φόρμα και επιτρέποντας στον χρήστη να υποβάλει περισσότερα στοιχεία του υπαλλήλου, αν χρειάζεται.

Αν τα δεδομένα της φόρμας αποτύχουν στην επικύρωση ή αν πρόκειται για ένα αίτημα GET, απλώς αποδίδεται το πρότυπο employee_form.html μαζί με το αντικείμενο της φόρμας και ένα κενό μήνυμα.

```python
@app.route("/employee/<int:employee_id>", methods=["GET", "POST"])
def show_employee_form_update(employee_id):
    message = ""
    employee = Employee.query.filter(Employee.id == employee_id).first()
    if not employee:
        return render_template("404.html", title="404"), 404
    form = EmployeeForm(obj=employee)
    if form.validate_on_submit():
        name = form.name.data
        surname = form.surname.data
        birth_year = form.birth_year.data
        email = form.email.data
        phone = form.phone.data
        job_position = form.job_position.data
        employee.name = name
        employee.surname = surname
        employee.birth_year = birth_year
        employee.email = email
        employee.phone = phone
        employee.job_position = job_position
        db_session.commit()
        return redirect(url_for("show_employees"))
    return render_template("employee_form.html", form=form, message=message, employee=employee)
```

Το παραπάνω κομμάτι του κώδικα χειρίζεται αιτήσεις για την ενημέρωση των πληροφοριών ενός υφιστάμενου υπαλλήλου. Δέχεται τόσο αιτήσεις **GET** όσο και αιτήσεις **POST**. Όταν γίνεται αίτηση **GET**, αντλεί τις πληροφορίες του υπαλλήλου από τη βάση δεδομένων με βάση το παρεχόμενο **employee_id**. Εάν ο υπάλληλος δεν υπάρχει, εμφανίζει μια σελίδα σφάλματος 404. Ωστόσο στη περίπτωση που ο υπάλληλος υπάρχει, αρχικοποιεί το **EmployeeForm** με τα δεδομένα του υπαλλήλου (**obj=employee**), επιτρέποντας την αυτόματη συμπλήρωση των πεδίων της φόρμας με τις υπάρχουσες πληροφορίες.

Σε μια αίτηση POST (συνήθως μετά την υποβολή της φόρμας), πρώτα επικυρώνει τα δεδομένα της φόρμας. Εάν τα δεδομένα της φόρμας περάσουν την επικύρωση, εξάγει τα ενημερωμένα στοιχεία του υπαλλήλου από τα πεδία της φόρμας και ενημερώνει το αντίστοιχο **employee** με αυτά τα νέα στοιχεία. Στη συνέχεια, δεσμεύει τις αλλαγές στη βάση δεδομένων χρησιμοποιώντας την **db_session.commit()** και ανακατευθύνει τον χρήστη πίσω στο /**employees**, το οποίο εμφανίζει όλους τους υπαλλήλους

Στη περίπτωση τώρα που τα δεδομένα της φόρμας αποτύχουν στην επικύρωση ή αν πρόκειται για αίτηση **GET**, εμφανίζει το πρότυπο **employee_form.html** μαζί με το αντικείμενο της φόρμας και το **employee**.

Αντίστοιχα για τις αιτήσεις των εργαζομένων

```
@app.route("/requests", methods=["GET"])
def show_requests():
    leave_requests = LeaveRequest.query.all()
    return render_template("requests.html", leave_requests=leave_requests)

@app.route('/request/<uid>', methods=['GET','POST'])
def show_request_form(uid):
    message = ""
    employee = Employee.query.filter(Employee.id == uid).first()
    form = LeaveRequestForm(employee=employee)

    if form.validate_on_submit():
        start_date = form.start_date.data
        end_date = form.end_date.data
        reason = form.reason.data

        request = LeaveRequest(
            start_date=start_date,
            end_date=end_date,
            reason=reason,
            employee=employee
        )
        db_session.add(request)
        db_session.commit()

        leave_requests = employee.leave_requests

        flash('Record was successfully added')

        return render_template("employee_requests.html", employee=employee, leave_requests=leave_requests)
    return render_template("employee_form.html", form=form, message=message, employee=employee)
```

```
@app.route("/request/<uid>/<jid>/delete", methods = ["GET"])
def delete_request(uid, jid):
    employee = Employee.query.filter(Employee.id == uid).first()
    leaveRequest = LeaveRequest.query.filter(LeaveRequest.id == jid).first()
    db_session.delete(leaveRequest)
    employee.leave_requests.remove(leaveRequest)
    db_session.commit()
    leave_requests = employee.leave_requests
    return render_template("employee_requests.html", employee=employee, leave_requests=leave_requests)
```

Το παραπάνω κομμάτι του κώδικα δέχεται αιτήσεις **GET** και χειρίζεται τη διαγραφή μιας αίτησης άδειας που σχετίζεται με έναν συγκεκριμένο υπάλληλο. Κατά την πραγματοποίηση ενός **GET** αιτήματος γίνεται ανάκτηση του αιτήματος άδειας με βάση το employee id αναγνωριστικό του υπαλλήλου και το request id αναγνωριστικό αιτήματος άδειας που παρέχονται αντίστοιχα. Ύστερα, διαγράφεται το αίτημα άδειας

από τον κατάλογο των αιτήσεων που σχετίζονται με τον υπάλληλο καλώντας την **employee.leave_requests.remove(leaveRequest).**

Εφόσον πραγματοποιηθούν αυτές οι αλλαγές στη βάση δεδομένων, δεσμεύονται στη βάση με **db_session.commit()**.

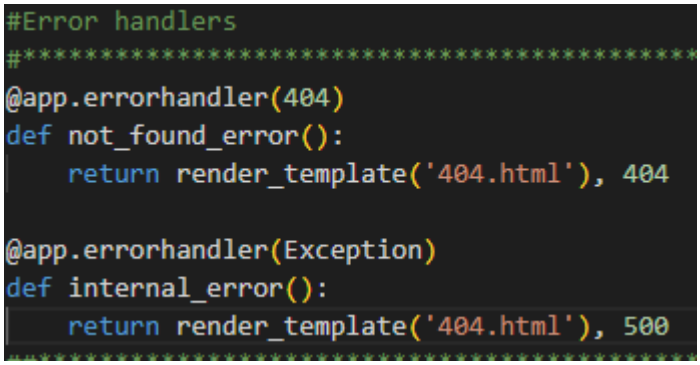
Εν κατακλείδι, ανακτά τον ενημερωμένο κατάλογο αιτήσεων άδειας για τον υπάλληλο και εμφανίζει το πρότυπο **employee_requests.html**, περνώντας το **employee** και τον ενημερωμένο κατάλογο αιτήσεων άδειας ως μεταβλητές περιβάλλοντος

```python
@app.route("/requests/<employee_id>", methods=["GET"])
def show_employee_requests(employee_id):
    employee = Employee.query.filter(Employee.id == employee_id).first()
    requests = employee.leave_requests
    return render_template("employee_requests.html", employee=employee, requests=requests)
```

```python
@app.route("/request/<jid>", methods=["DELETE"])
def delete_request_by_id(jid):
    request = LeaveRequestForm.query.filter(LeaveRequestForm.id == jid).first()
    employee = LeaveRequestForm.employee

    print(request)
    db_session.delete(request)
    db_session.commit()
    requests = employee.requests
    return render_template("employee_requests.html", employee=employee, requests=requests)
```

Για τα επόμενα κομμάτια το κώδικα, η πρώτη διαδρομή **/requests/<employee_id>** είναι υπεύθυνη για την εμφάνιση όλων των αιτήσεων αδειών που σχετίζονται με έναν συγκεκριμένο υπάλληλο. Η δεύτερη διαδρομή **/request/<jid>** είναι υπεύθυνη για τη διαγραφή ενός αιτήματος άδειας με βάση το αναγνωριστικό του **id.**



Στο παραπάνω μέρος εμφανίζονται και οι δύο συναρτήσεις χειρισμού σφαλμάτων.

Η πρώτη χειρίζεται τα σφάλματα **HTTP 404**, τα οποία εμφανίζονται όταν το ζητούμενο δεν βρέθηκε.

Η δεύτερη χειρίζεται τυχόν μη καταγεγραμμένες παρεκκλίσεις που εμφανίζονται εντός της εφαρμογής.

3.1 Κώδικας στο forms.py

forms.py > ...

```python
from flask_wtf import FlaskForm, CSRFProtect
from wtforms import StringField, SubmitField, IntegerField, EmailField, DateField
from wtforms.validators import DataRequired, Length, Email

class EmployeeForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired(), Length(3, 15)])
    surname = StringField('Surname', validators=[DataRequired(), Length(3, 20)])
    birth_year = IntegerField('Birth Year', validators=[DataRequired()])
    email = EmailField('Email', validators=[DataRequired(), Email()])
    phone = StringField('Phone', validators=[DataRequired()])
    job_position = StringField('Job Position', validators=[DataRequired()])
    submit = SubmitField('Submit')


class LeaveRequestForm(FlaskForm):
    start_date = DateField('Start Date', validators=[DataRequired()])
    end_date = DateField('End Date', validators=[DataRequired()])
    reason = StringField('Reason', validators=[DataRequired(), Length(50, 180)])
    submit = SubmitField('Submit')
```

Κλάση: **EmployeeForm**

- Κληρονομεί από την FlaskForm, υποδεικνύοντας ότι είναι μια φόρμα WTForms σχεδιασμένη να λειτουργεί με την Flask.
- Ορίζει πεδία για διάφορα στοιχεία του εργαζομένου, όπως όνομα, επώνυμο, έτος γέννησης, ηλεκτρονικό ταχυδρομείο, τηλέφωνο και θέση εργασίας.
- Κάθε πεδίο συνδέεται με validators που επιβάλλουν ορισμένες απαιτήσεις (π.χ., τα δεδομένα πρέπει να είναι παρόντα, όρια μήκους, έγκυρη μορφή ηλεκτρονικού ταχυδρομείου).
- Το πεδίο υποβολής είναι ένα κουμπί που χρησιμοποιείται για την υποβολή της φόρμας.

Κλάση: **LeaveRequestForm**

- Περιλαμβάνει πεδία για την ημερομηνία έναρξης, την ημερομηνία λήξης και τον λόγο της αίτησης άδειας.
- Οι validators διασφαλίζουν ότι παρέχονται τα απαιτούμενα δεδομένα και ότι ο λόγος εμπίπτει σε ένα συγκεκριμένο εύρος μήκους.
- Όπως και το **EmployeeForm**, περιλαμβάνει ένα κουμπί υποβολής για την υποβολή της φόρμας.

Οι παραπάνω κλάσεις παρέχουν έναν τρόπο για τον ορισμό φορμών HTML σε εφαρμογές Flask, διευκολύνοντας τον χειρισμό της επικύρωσης και της επεξεργασίας δεδομένων φόρμας.

3.2 Κώδικας στο forms.py

Ο παρακάτω κώδικας είναι υπεύθυνος για τον προσδιορισμό των μοντέλων **SQLAlchemy** για την διαχείριση των δεδομένων των εργαζομένων και των αιτήσεων άδειας σε μια βάση δεδομένων.

```
#Ορισμός βασικής κλάσης για όλε
class Base(DeclarativeBase):
    pass
```

Αρχικά, ορίζεται η Base κλάση η οποία χρησιμεύει ως βάση για όλα τα μοντέλα SQLAlchemy και παρέχει κοινή λειτουργικότητα για την αντιστοίχιση αντικειμένου με την ORM.

```
#Κλάση για τα στοιχεία των employees
class Employee(Base):
    __tablename__ = 'employees'
    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(30), nullable=False,)
    surname: Mapped[str] = mapped_column(String(50), nullable=False)
    birth_year: Mapped[int] = mapped_column(Integer, nullable=False)
    email: Mapped[str] = mapped_column(String(50), unique=True, nullable=False)
    phone: Mapped[str] = mapped_column(String(15), nullable=False)
    job_position: Mapped[str] = mapped_column(String(50), nullable=False)
    date_created: Mapped[datetime] = mapped_column(DateTime(timezone=True), default=datetime.now)

    #Διόρθωση βρόχου ανακατεύθυνσης στη διαδρομή Flask για την υποβολή αιτήσεων άδειας
    leave_requests : Mapped[List["LeaveRequest"]] = relationship("LeaveRequest", back_populates="employee")

    def __str__(self):
        return f"<Employee id={self.id}, name={self.name}, surname={self.surname}, birth_year={self.birth_yea
```

Κλάση εργαζομένων:

- Αντιπροσωπεύει τα δεδομένα των εργαζομένων που είναι αποθηκευμένα στον πίνακα **employees**.
- Τα χαρακτηριστικά περιλαμβάνουν τα id, name, surname, birth_year, email, phone, job_position και date_created.
- Ορίζει μια σχέση ένα προς πολλά με την κλάση **LeaveRequest** για το χειρισμό αιτήσεων άδειας που σχετίζονται με έναν υπάλληλο.

```python
class LeaveRequest(Base):
    __tablename__ = 'leaverequests'
    id: Mapped[int] = mapped_column(primary_key=True)
    start_date: Mapped[Date] = mapped_column(Date)
    end_date: Mapped[Date] = mapped_column(Date)
    reason : Mapped[str] = mapped_column(String(20), nullable=False)
    status : Mapped[str] = mapped_column(String(50), default='Pending')
    employee_id : Mapped[int] = mapped_column(ForeignKey('employees.id'))
    date_created: Mapped[datetime] = mapped_column(DateTime(timezone=True), default=datetime.now)

    #Ορισμός αμφίδρομης σχέσης μεταξύ των κλάσεων Employee και LeaveRequest
    employee : Mapped[List["Employee"]] = relationship("Employee", back_populates="leave_requests")

    def __str__(self):
        return f"<start_date={self.start_date}, end_date={self.end_date}, reason={self.reason}, status={self.status}>"
```

Κλάση αδειών:

- Αντιπροσωπεύει τα αιτήματα άδειας που είναι αποθηκευμένα στον πίνακα **leaverequests**.
- Τα χαρακτηριστικά περιλαμβάνουν id, start_date, end_date, reason, status, employee_id και date_created.
- Ορίζει μια σχέση πολλά προς ένα με την κλάση **Employee** για τη συσχέτιση αιτήσεων άδειας με υπαλλήλους.

Οι παραπάνω κλάσεις, χρησιμεύουν ως σχεδιαγράμματα για την οργάνωση και τη διαχείριση των δεδομένων στη βάση δεδομένων.

4. Συζήτηση

Αρχικά, ξεκίνησα να ετοιμάζω την βάση από την αρχή αλλά γρήγορα κατάλαβα ότι δεν μου φτάνει ο χρόνος να την ολοκληρώσω, έπεφτα συνέχεια σε error έλυνα το ένα πίσω από το άλλο και μετά έβγαιναν και άλλα στη φόρα. Ωστόσο έπρεπε κάτι να παραδώσω, γι' αυτό λοιπόν έγινε αποδόμησή της δική σας βάση που είχατε κάνει στο μάθημα και προσπάθησα να τη μιμηθώ από την αρχή έχοντας πλέον ως οδηγό το δικό σας πάτημα. Ακόμα και εκεί τα προβλήματα δεν έλειψαν, κάτι κεφαλαία κάτι πεζά κάτι αποστάσεις κλπ με έκαψαν.

Ακόμη, κάτι που δεν έχω βρει σίγουρα είναι μπροστά στα μάτια μου αλλά δεν το βλέπω, κάτι έχω κάνει και δεν μου φέρνει το σύνολο των αδειών στο /requests.

Θα ήθελα στις θέσεις εργασίας να βάλω options έτσι να υπάρχουν επιλογές. Ακόμη θα ήθελα να βάλω στο status της άδειας pending ως default τιμή και εκεί επίσης επιλογές δεκτή και απορριφθείσα.

Συνολικά, τρομερή εργασία έχω πάθει σοκ με το πως λειτουργούν όλα τόσο αρμονικά.