

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ



**ΔΙΑΧΕΙΡΙΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ
ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ
ΑΕΡΟΠΟΡΙΚΩΝ ΕΤΑΙΡΙΩΝ**

Νικόλαος Πολυχρονόπουλος

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων
για την απόκτηση του Διπλώματος
Μηχανολόγου Μηχανικού

Βόλος, 2024

**ΔΙΑΧΕΙΡΙΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ
ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ
ΑΕΡΟΠΟΡΙΚΩΝ ΕΤΑΙΡΙΩΝ**

Διπλωματική εργασία
του Νικόλαου Πολυχρονόπουλου

Επιβλέπων Καθηγητής: Γεώργιος Κοζανίδης

Τριμελής Εξεταστική Επιτροπή

- | | |
|----------------------|--|
| 1 ^ο Μέλος | Γεώργιος Κοζανίδης , Αναπληρωτής Καθηγητής,
Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας |
| 2 ^ο Μέλος | Γεώργιος Λυμπερόπουλος , Καθηγητής,
Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας |
| 3 ^ο Μέλος | Γεώργιος Σαχαρίδης , Αναπληρωτής Καθηγητής,
Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας |

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις ευχαριστίες μου σε όλους όσους συνέβαλαν στην εκπόνησή της. Ευχαριστώ θερμά τον επιβλέπων καθηγητή μου, κ. Γεώργιο Κοζανίδη, για την επιστημονική του καθοδήγηση, τις υποδείξεις του, τη συνεχή του υποστήριξη και το ενδιαφέρον που έδειξε από την αρχή μέχρι το τέλος. Επίσης, ευχαριστώ τα μέλη της εξεταστικής επιτροπής Γεώργιο Λυμπερόπουλο και Γεώργιο Σαχαρίδη που αφιέρωσαν χρόνο στη μελέτη της εργασίας μου.

Τελειώνοντας, θα ήταν παράλειψη μου αν δεν ευχαριστήσω θερμά την οικογένεια μου και τους φίλους μου που στάθηκαν δίπλα μου τόσο κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας όσο και καθ' όλη τη διάρκεια των σπουδών μου στο Πανεπιστήμιο Θεσσαλίας.

Πολυχρονόπουλος Νίκος

Περίληψη

Η διπλωματική εργασία παρουσιάζει τις μεθοδολογίες που χρησιμοποιούνται για τη βελτιστοποίηση της διαχείρισης αδειών και της κατανομής πληρώματος στις αεροπορικές εταιρείες. Παρουσιάζεται η εφαρμογή που αναπτύχθηκε σε `wpf .net` για την επίλυση των προβλημάτων βελτιστοποίησης, τη διαχείριση βάσης δεδομένων `sql server` με φιλικό προς το χρήστη γραφικό περιβάλλον για την εισαγωγή όλων των απαιτούμενων δεδομένων των βελτιστοποιήσεων και δυναμικής παρουσίασης των αποτελεσμάτων. Στο πρώτο κεφάλαιο παρουσιάζεται ο σκοπός και η δομή της διπλωματικής εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζονται μια πληθώρα προβλημάτων που επιλύονται μέσω του κλάδου της επιχειρησιακής έρευνας και επίσης παρουσιάζονται τα λογισμικά που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Στο τρίτο κεφάλαιο, παρουσιάζεται η διαδικασία ανάθεσης αδειών σύμφωνα με τις προτιμήσεις του προσωπικού, η μοντελοποίηση του προβλήματος και ο κώδικας επίλυσης σε `C#, Cplex`. Το τέταρτο κεφάλαιο ασχολείται με την αρχικοποίηση της τυχαίας ανάθεσης των υπολειπόμενων ημερών άδειας περιγράφοντας την μοντελοποίηση, τον κώδικα και υπολογιστικό παράδειγμα. Στο πέμπτο κεφάλαιο, εξετάζεται η αρχικοποίηση `Set-Cover` και `Set-Partition` για την κατανομή πληρώματος, με αναλυτική περιγραφή και υπολογιστικά παραδείγματα. Το έκτο κεφάλαιο είναι ένα εγχειρίδιο της εφαρμογής που περιγράφει τη δομή, τις βασικές λειτουργίες και τα εργαλεία βελτιστοποίησης που χρησιμοποιούνται. Τέλος, το έβδομο κεφάλαιο αναλύει την αρχιτεκτονική της εφαρμογής, εξηγώντας τη δομή της βάσης δεδομένων και τις `CRUD` εντολές που χρησιμοποιήθηκαν για τη διαχείριση των δεδομένων. Η εργασία καταλήγει με την ανάλυση της οθόνης `Vacation Planning` και τη διαδικασία εισαγωγής παραμέτρων για την επίλυση του προβλήματος ανάθεσης αδειών και της εξαγωγής της λύσης .

Abstract

This thesis presents methodologies used for optimizing leave management and crew rostering in airlines.

The application developed in WPF .NET is introduced for solving optimization problems, managing an SQL Server database with a user-friendly graphical interface for inputting all required optimization data and dynamically presenting the results.

The first chapter outlines the purpose and structure of the thesis.

The second chapter discusses a variety of problems solved through operations research and presents the software used for developing the application.

The third chapter details the process of assigning leaves according to staff preferences, the problem modeling, and the solution code in C#, Cplex.

The fourth chapter deals with the initialization of random assignment of remaining leave days, describing the modeling, code, and a computational example.

The fifth chapter examines the initialization of Set-Cover and Set-Partition for crew allocation, with detailed descriptions and computational examples.

The sixth chapter is a user manual of the application that describes its structure, basic functions, and the optimization tools used.

The seventh chapter analyzes the application architecture, explaining the database structure and the CRUD operations used for data management. The thesis concludes with the analysis of the Vacation Planning screen and the process of parameter input for solving the leave assignment problem and extracting the solution.

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή	1
1.1) Σκοπός.....	1
1.2) Δομή διπλωματικής.....	2
Κεφάλαιο 2: Επιχειρησιακή Έρευνα στις αεροπορικές εταιρίες, Ανάλυση λογισμικών	3
2.1) Πορεία του κλάδου της επιχειρησιακής έρευνας στις αεροπορικές εταιρίες	3
2.1) Παρουσίαση των λογισμικών της εφαρμογής	3
Κεφάλαιο 3: Ανάθεση αδειών σύμφωνα με τις προτιμήσεις του προσωπικού	4
3.1) Λογική Ανάθεσης	4
3.2) Ανάλυση προτιμήσεων άδειας υπαλλήλου	5
3.3) Μορφοποίηση του προβλήματος.....	8
3.4) Κώδικας βελτιστοποίησης σε C#, IBM CPLEX.....	11
3.4.1) Δείκτες, παράμετροι.....	11
3.4.2) Μεταβλητές απόφασης, Αντικειμενική Συνάρτηση.....	11
3.4.3) Περιορισμοί.....	12
3.4.4) Έλεγχος προτίμησης άδειας με βάση τη λογική ανάθεσης	15
3.4.5) Εξαγωγή αποτελεσμάτων	17
Κεφάλαιο 4: Αρχικοποίηση τυχαίας ανάθεσης υπολειπόμενων ημερών άδειας.....	18
4.1) Ανάλυση του προβλήματος ανάθεσης υπολειπόμενων ημερών αδειών	18
4.2) Μορφοποίηση του προβλήματος	18
4.3) Κώδικας βελτιστοποίησης σε C#, IBM CPLEX.....	19
4.3.1) Δείκτες, παράμετροι.....	19
4.3.2) Μεταβλητές απόφασης, Αντικειμενική Συνάρτηση.....	19
4.3.3) Περιορισμοί.....	20
4.3.4) Εξαγωγή αποτελεσμάτων	21
4.4) Υπολογιστικό παράδειγμα.....	22
4.4.1) Περιγραφή δεδομένων εισαγωγής	22
4.4.2) Αποτελέσματα: Αρχείο .lp	22
4.4.3) Αποτελέσματα: Αρχείο .sol.....	22
Κεφάλαιο 5 : Crew Rostering : Αρχικοποίηση Set-Cover, Set- Partition	24
5.1) Μορφοποίηση Set-Cover , Set-Partition	24
5.1.1) Set-Partition.....	24
5.1.2) Set-Cover	25
5.2) Κώδικας αρχικοποίησης Set-Cover, Set-Partition σε C#, IBM CPLEX.....	26
5.2.1) Αρχικοποίηση Set-Partiton	26
5.2.1.1) Αρχικοποίηση δεικτών, παραμέτρων.....	26
5.2.1.2) Δείκτες, παράμετροι	27
5.2.1.3) Μεταβλητές απόφασης, Αντικειμενική Συνάρτηση	27
5.2.1.4) Περιορισμοί.....	28
5.2.1.5) Εξαγωγή αποτελεσμάτων	29
5.2.1.6) Υπολογιστικό παράδειγμα	29
5.2.1.6.1) Αποτελέσματα: Αρχείο .lp	30
5.2.1.6.2) Αποτελέσματα: Αρχείο .sol	30
5.2.2) Αρχικοποίηση Cover	31
5.2.2.1) Αρχικοποίηση δεικτών, παραμέτρων	31
5.2.2.2) Δείκτες, παράμετροι	32
5.2.2.3) Μεταβλητές απόφασης, Αντικειμενική Συνάρτηση	32
5.2.2.4) Περιορισμοί.....	33
5.2.2.5) Εξαγωγή αποτελεσμάτων	33
5.2.2.6) Υπολογιστικό παράδειγμα	34
5.2.2.6.1) Αποτελέσματα: Αρχείο .lp.....	34
5.2.2.6.2) Αποτελέσματα: Αρχείο .sol	34

Κεφάλαιο 6: Εγχειρίδιο εφαρμογής.....	35
6.1) Δομή, λειτουργίες εφαρμογής	35
6.2) Αρχείο υπαλλήλων.....	39
6.2.1) 1 ^η Καρτέλα Αρχείου υπαλλήλων: Γενικές Πληροφορίες	39
6.2.2) 2 ^η Καρτέλα Αρχείου υπαλλήλων: Προτιμήσεις αδειών υπαλλήλου	40
6.3) Αρχείο χρονοδιαγράμματος απαιτήσεων	43
6.3.1) 1 ^η Καρτέλα Αρχείου χρονοδιαγράμματος: Γενικές Πληροφορίες	43
6.3.2) 2 ^η Καρτέλα Αρχείου χρονοδιαγράμματος: Χρονοδιάγραμμα	44
6.4) Αρχείο προγραμματισμού διακοπών	45
6.4.1) 1 ^η Καρτέλα προγραμματισμού διακοπών: Γενικές Πληροφορίες	45
6.4.2) 2 ^η Καρτέλα προγραμματισμού διακοπών: Αποτελέσματα	47
6.4.2.1) 1 ^η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Yij	48
6.4.2.2) 2 ^η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Yijrz	49
6.4.2.3) 3 ^η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Xit.....	50
6.4.2.4) 4 ^η Υποκαρτέλα Αποτελεσμάτων: Entitlement	51
6.4.2.5) 5 ^η Υποκαρτέλα Αποτελεσμάτων: RLLt Plan	52
6.4.2.6) 6 ^η Υποκαρτέλα Αποτελεσμάτων: Column Generation Results	52
6.5) Αρχείο χωρών, νομών, πόλεων	53
6.5.1) Αρχείο χωρών, νομών, πόλεων : Περιγραφή	53
6.6) Αρχείο Αεροδρομίων	55
6.6.1) 1 ^η Καρτέλα Αρχείου αεροδρομίων : Γενικές Πληροφορίες	55
6.7) Αρχείο πτήσεων	56
6.7.1) 1 ^η Καρτέλα Αρχείου πτήσεων: Γενικές Πληροφορίες.....	56
6.8) Αρχείο διαδρομών	57
6.8.1) 1 ^η Καρτέλα Αρχείου διαδρομών: Γενικές Πληροφορίες.....	57
6.9) Αρχείο κατανομής πληρώματος	58
6.9.1) 1 ^η Καρτέλα αρχείου κατανομής πληρώματος: Γενικές Πληροφορίες.....	58
6.9.2) 2 ^η Καρτέλα αρχείου κατανομής πληρώματος : Αποτελέσματα	60
Κεφάλαιο 7: Ανάλυση αρχιτεκτονικής της εφαρμογής	61
7.1) Αρχιτεκτονική, MVVM Θεωρία	61
7.2) Αρχιτεκτονική, MVVM Υλοποίηση.....	62
7.2.1) Οργάνωση προγραμματιστικού περιβάλλοντος.....	62
7.2.2) Βάση Δεδομένων	64
7.2.2.1) Βάση Δεδομένων: Θεωρία	62
7.2.2.2) Βάση Δεδομένων: Σχεσιακό Διάγραμμα	65
7.2.2.3) Βάση Δεδομένων: Entities	66
7.2.3) Model	71
7.2.3.1) Model: Θεωρία	71
7.2.3.2) Model: Κλάσεις	72
7.2.4) Εντολές διαχείρισης δεδομένων / CRUD Commands	74
7.2.4.1) Get Command	74
7.2.4.2) Add, Save Commands	77
7.2.5) View Model	80
7.3) Vacation Planning	82
7.3.1) Vacation Planning, Ανάλυση Model	82
7.3.2) Vacation Planning, Ανάλυση View Model	86
7.3.3) Vacation Planning Optimization, Δυναμική εισαγωγή παραμέτρων	89
7.3.4) Vacation Planning Optimization, Εξαγωγή δεδομένων βέλτιστης λύσης	91
Κεφάλαιο 8: Συμπεράσματα, Βιβλιογραφία	97
8.1) Συμπεράσματα	97
8.2) Βιβλιογραφία	97

1.1) Σκοπός

Ο σκοπός της διπλωματικής εργασίας είναι να αναδείξει τα πλεονεκτήματα που προσφέρει ένα σύστημα διαχείρισης αδειών (Vacation Management System, VMS) με φιλικό προς τον χρήστη περιβάλλον και βάση δεδομένων στον τομέα της επιχειρησιακής έρευνας. Η εφαρμογή αυτή αποσκοπεί στη βελτίωση της αποδοτικότητας και της οργάνωσης των διαδικασιών ανάθεσης αδειών και κατανομής πληρώματος, αξιοποιώντας προηγμένες τεχνικές βελτιστοποίησης.

Ο στόχος ήταν να εισαγάγουμε δυναμικά τις παραμέτρους της βελτιστοποίησης μέσω της βάσης δεδομένων. Αυτή η προσέγγιση επιτρέπει μεγαλύτερη ευελιξία, αυξάνει την ακρίβεια και διευκολύνει την ενημέρωση και τη διαχείριση των δεδομένων σε πραγματικό χρόνο. Χρησιμοποιώντας μια βάση δεδομένων, μπορούμε να διασφαλίσουμε ότι οι πληροφορίες είναι πάντα επικαιροποιημένες και προσβάσιμες, ενώ παράλληλα μειώνουμε την πιθανότητα σφαλμάτων που μπορεί να προκύψουν από τη χειροκίνητη εισαγωγή δεδομένων.

Το VMS ενσωματώνει σύγχρονες μεθόδους και τους optimization solvers IBM CPLEX και Gurobi, τα οποία επιτρέπουν την αποτελεσματική επίλυση των βελτιστοποιήσεων. Η φιλική προς τον χρήστη εφαρμογή διευκολύνει τη διαχείριση των δεδομένων και την εισαγωγή παραμέτρων, ενώ η δυναμική εξαγωγή των δεδομένων βέλτιστης λύσης διασφαλίζει ότι οι προτεινόμενες λύσεις είναι πάντα ενημερωμένες και βέλτιστες.

Η εφαρμογή αυτή παρέχει στους χρήστες τη δυνατότητα να διαχειρίζονται εύκολα τις προτιμήσεις άδειας του προσωπικού, να αναλύουν τις απαιτήσεις χρονοδιαγράμματος και να βελτιστοποιούν την κατανομή των αδειών με βάση προκαθορισμένους περιορισμούς και προτιμήσεις. Μέσω της χρήσης προηγμένων αλγορίθμων, το VMS εξασφαλίζει ότι οι άδειες χορηγούνται δίκαια και αποδοτικά, μειώνοντας ταυτόχρονα το κόστος για την εταιρία.

Επιπλέον, η εφαρμογή παρέχει λεπτομερή αναφορά των αποτελεσμάτων και των παραμέτρων που χρησιμοποιήθηκαν, διευκολύνοντας τη λήψη αποφάσεων και την παρακολούθηση της απόδοσης του συστήματος.

Η ανάπτυξη της εφαρμογής βασίστηκε σε σύγχρονες πρακτικές ανάπτυξης λογισμικού, διασφαλίζοντας την αξιοπιστία και τη σταθερότητά της. Η υιοθέτηση της αρχιτεκτονικής MVVM διευκολύνει τη συντήρηση και την επεκτασιμότητα του κώδικα, επιτρέποντας τη συνεχή βελτίωση και αναβάθμιση της εφαρμογής. Η χρήση προηγμένων εργαλείων βελτιστοποίησης, όπως το IBM CPLEX, διασφαλίζει την υψηλή απόδοση και ακρίβεια των αποτελεσμάτων.

1.2) Δομή Διπλωματικής

Η διπλωματική εργασία είναι δομημένη σε οκτώ κεφάλαια.

Το δεύτερο κεφάλαιο περιγράφει τα εργαλεία λογισμικού και τις μεθόδους που χρησιμοποιήθηκαν για την επίλυση των προβλημάτων βελτιστοποίησης, όπως το IBM CPLEX και η γλώσσα προγραμματισμού C# και γίνεται μια σύντομη περιγραφή διαφόρων προβλημάτων επιχειρησιακής έρευνας στις αεροπορικές εταιρίες.

Στο τρίτο κεφάλαιο, παρουσιάζεται λεπτομερώς η διαδικασία ανάθεσης αδειών σύμφωνα με τις προτιμήσεις του προσωπικού, περιλαμβάνοντας τη λογική ανάθεσης, την ανάλυση των προτιμήσεων άδειας, τη μοντελοποίηση του προβλήματος και τον κώδικα βελτιστοποίησης.

Το τέταρτο κεφάλαιο ασχολείται με την αρχικοποίηση της τυχαίας ανάθεσης των υπολειπόμενων ημερών άδειας, περιγράφοντας τη μεθοδολογία, τη μοντελοποίηση του προβλήματος, τον κώδικα βελτιστοποίησης και ένα υπολογιστικό παράδειγμα με δεδομένα εισαγωγής και αποτελέσματα.

Στο πέμπτο κεφάλαιο, εξετάζεται η αρχικοποίηση Set-Cover και Set-Partition για το πρόβλημα κατανομής πληρώματος. Παρέχεται αναλυτική περιγραφή της μεθοδολογίας, κώδικας και υπολογιστικά παραδείγματα για κάθε προσέγγιση.

Το έκτο κεφάλαιο αποτελεί το εγχειρίδιο της εφαρμογής που περιγράφει τη δομή και βασικές λειτουργίες της, συμπεριλαμβανομένων όλων των αρχείων δεδομένων, και των εργαλείων βελτιστοποίησης. Εξηγεί επίσης εκτενώς πως η εφαρμογή διαχειρίζεται τα προβλήματα βελτιστοποίησης που αναλύονται στα κεφάλαια 3, 4 και 5 .

Το έβδομο κεφάλαιο αναλύει την αρχιτεκτονική της εφαρμογής MVVM (Model-View-View Model) . Εξηγεί τη δομή της βάσης δεδομένων και τις CRUD (Create, Read, Update, Delete) εντολές που χρησιμοποιήθηκαν για τη διαχείριση των δεδομένων. Το κεφάλαιο 7 επικεντρώνεται στην ανάλυση της οθόνης Vacation Planning, περιγράφει τη διαδικασία δυναμικής εισαγωγής παραμέτρων και εξαγωγής δεδομένων βέλτιστης λύσης για το πρόβλημα ανάθεσης αδειών που αναλύεται στο κεφάλαιο 3.

Τέλος, στο κεφάλαιο οκτώ παρουσιάζονται τα διάφορα συμπεράσματα που εξάχθηκαν με την ολοκλήρωση της παρούσας εργασίας. Επίσης, περιέχεται η βιβλιογραφία που χρησιμοποιήθηκε.

2.1) Η Πορεία του Κλάδου της Επιχειρησιακής Έρευνας στον Τομέα των Αερομεταφορών

Η επιχειρησιακή έρευνα έχει διαδραματίσει καθοριστικό ρόλο στον τομέα των αερομεταφορών, βοηθώντας στην επίλυση πολύπλοκων προβλημάτων που σχετίζονται με τη διαχείριση και βελτιστοποίηση λειτουργιών. Ακολουθούν παραδείγματα μιας ποικιλίας προβλημάτων που επιλύονται στον κλάδο των αερομεταφορών με τη βοήθεια της επιχειρησιακής έρευνας όπως:

A) Ανάθεση Αδειών (Vacation Planning)

Η ανάθεση αδειών είναι κρίσιμη για τη διαχείριση του ανθρώπινου δυναμικού. Χρησιμοποιούνται αλγόριθμοι βελτιστοποίησης για να προγραμματιστούν οι άδειες των εργαζομένων, διασφαλίζοντας ότι η λειτουργία της εταιρείας παραμένει απρόσκοπτη ενώ οι εργαζόμενοι ικανοποιούνται.

B) Κατανομή Πληρώματος (Crew Rostering)

Η κατανομή πληρώματος αφορά τη δημιουργία προγραμμάτων εργασίας που συμμορφώνονται με τους κανονισμούς και τις προτιμήσεις του πληρώματος. Μέσω μαθηματικών μοντέλων, επιτυγχάνεται η βέλτιστη κατανομή του πληρώματος, μειώνοντας το κόστος και βελτιώνοντας την απόδοση.

Γ) Βελτιστοποίηση Δρομολογίων (Flight Scheduling)

Η βελτιστοποίηση δρομολογίων περιλαμβάνει τη δημιουργία χρονοδιαγραμμάτων πτήσεων που μεγιστοποιούν την αποδοτικότητα και την πληρότητα των αεροσκαφών, ενώ ελαχιστοποιούν τις καθυστερήσεις. Αναπτύσσονται μοντέλα που λαμβάνουν υπόψη τη ζήτηση, τους περιορισμούς των αεροδρομίων και τη διαθεσιμότητα των αεροσκαφών.

Δ) Διαχείριση Εισιτηρίων (Revenue Management)

Η διαχείριση εισιτηρίων χρησιμοποιεί αλγόριθμους που προβλέπουν τη ζήτηση και ρυθμίζουν τις τιμές των εισιτηρίων δυναμικά για τη μεγιστοποίηση των εσόδων. Αυτό περιλαμβάνει την προσαρμογή των τιμών ανάλογα με την εποχή, την ημέρα και την πληρότητα των πτήσεων. Τέλος παρατίθεται η βιβλιογραφία και ο κατάλογος εικόνων και πινάκων που χρησιμοποιήθηκαν.

2.2) Παρουσίαση Λογισμικών της εφαρμογής

Visual Studio, WPF (.NET)

Το Visual Studio αποτέλεσε το βασικό περιβάλλον ανάπτυξης της εφαρμογής. Με την υποστήριξη του .NET Framework, η εφαρμογή αναπτύχθηκε χρησιμοποιώντας γλώσσες προγραμματισμού όπως η C# και τεχνολογίες όπως η XAML για την ανάπτυξη των γραφικών. Η Windows Presentation Foundation (WPF) προσφέρει τα απαραίτητα εργαλεία και βιβλιοθήκες για τη δημιουργία ολοκληρωμένων Full-stack εφαρμογών με τοπική βάση δεδομένων.

Syncfusion

Η βιβλιοθήκη Syncfusion χρησιμοποιήθηκε για τη δημιουργία πινάκων δεδομένων και ημερολογίων. Αυτή η βιβλιοθήκη παρέχει ισχυρά εργαλεία για την εύκολη διαχείριση και απεικόνιση δεδομένων.

SQL Server

Ο SQL Server χρησιμοποιήθηκε ως το κύριο σύστημα διαχείρισης βάσης δεδομένων. Παρέχει αξιόπιστες δυνατότητες αποθήκευσης και διαχείρισης δεδομένων, επιτρέποντας την αποδοτική ανάκτηση και επεξεργασία πληροφοριών, απαραίτητων για την εφαρμογή.

Cplex, Gurobi

Τα εργαλεία βελτιστοποίησης Cplex και Gurobi χρησιμοποιήθηκαν για την επίλυση των 3 προβλημάτων βελτιστοποίησης που αναλύονται στη διπλωματική. Συνεπώς η εφαρμογή μας επιτρέπει να λύσουμε τις μοντελοποιήσεις με όποιον solver προτιμάμε και ταυτόχρονα να συγκρίνουμε την απόδοση της βελτιστοποίησης ανά λογισμικό

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

Αποτελεί η υποχρέωση κάθε αεροπορικής εταιρίας να προσπαθήσει να χορηγήσει τις προτεινόμενες άδειες των υπαλλήλων της όσο πιο αποτελεσματικά γίνεται. Κάθε υπάλληλος συμπληρώνει μια ειδική φόρμα προτιμήσεων άδειας και κατάλληλο λογισμικό όπου επεξεργάζεται τις προτιμήσεις και χορηγεί όσες άδειες μπορεί με βάση ορισμένους περιορισμούς που αναλύονται στο κεφάλαιο (3.3).

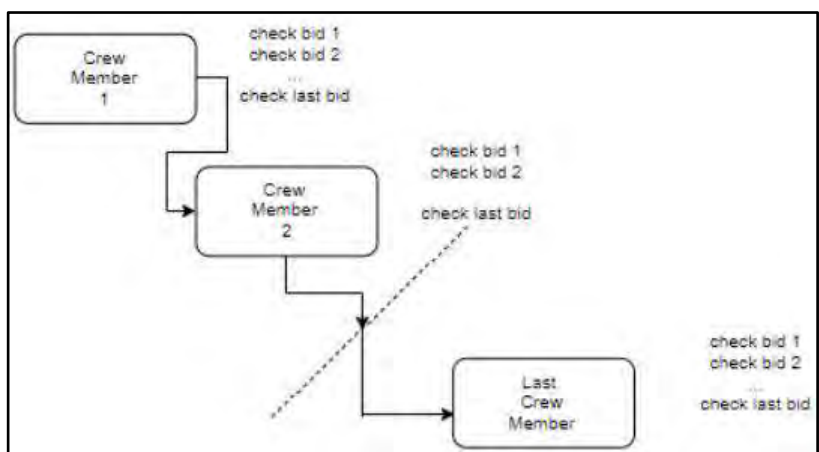
3.1) Λογική Ανάθεσης

Συνηθίζεται προτεραιότητα να δίνεται στους υπαλλήλους με τα περισσότερα χρόνια προϋπηρεσίας για την χορήγηση των προτιμήσεων τους για άδεια. Υπάρχουν 3 είδη διαφορετικών πολιτικών ανάθεσης αδειών που αναλύονται παρακάτω :

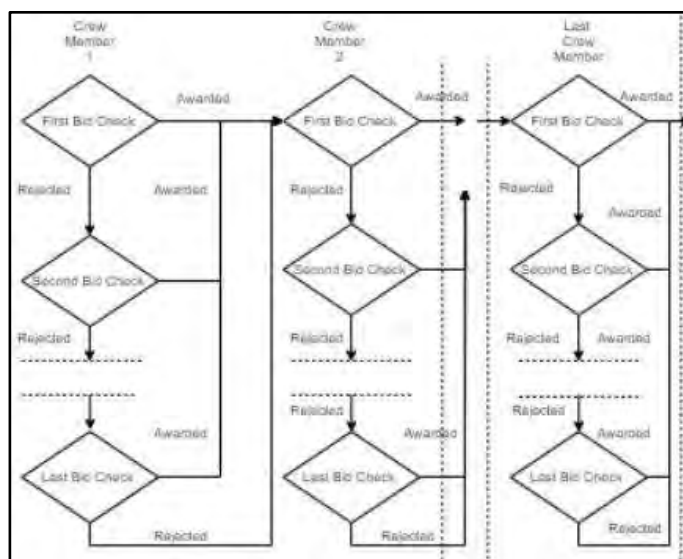
Strict Seniority: Η ανάθεση αδειών ξεκινά από τα παλαιότερα μέλη και καταλήγει στα νεότερα. Πρώτα ικανοποιούνται οι προτιμήσεις του μέλους με τα περισσότερα χρόνια υπηρεσίας και στη συνέχεια η διαδικασία επαναλαμβάνεται για το επόμενο μέλος σε προϋπηρεσία.

Fair Priority: Η ανάθεση ξεκινά από το αρχαιότερο μέλος, ελέγχοντας τις επιθυμίες του μέχρι να βρεθεί η πρώτη εφικτή προτίμηση, η οποία του ανατίθεται. Στη συνέχεια, ελέγχονται οι επιθυμίες των υπόλοιπων μελών με φθίνουσα σειρά προϋπηρεσίας. Μόλις και το νεότερο μέλος ικανοποιήσει μία από τις προτιμήσεις του, η διαδικασία ξεκινά από την αρχή με το αρχαιότερο μέλος.

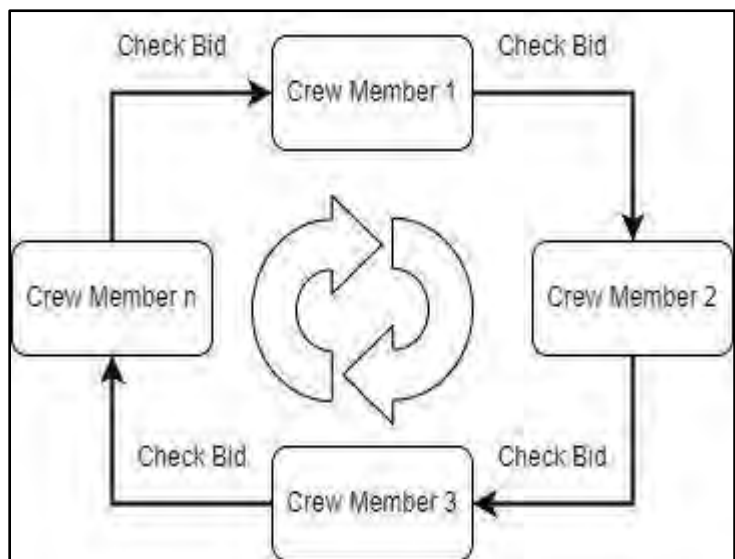
Bid by Bid: Η ανάθεση ξεκινά από το αρχαιότερο μέλος και συνεχίζεται στο επόμενο, ανεξάρτητα από το αν χορηγείται η πρώτη άδεια. Εξετάζουμε τις αιτήσεις άδειας κάθε υπαλλήλου με τη σειρά, μεταβαίνοντας από το ένα μέλος στο επόμενο μέχρι να εξεταστούν όλες οι αιτήσεις.



Εικόνα 3.1: Strict Seniority



Εικόνα 3.2: Fair Priority



Εικόνα 3.3: Bid by Bid

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

3.2) Ανάλυση προτιμήσεων αδείας υπαλλήλου

Ο υπάλληλος μπορεί να εισάγει τρεις διαφορετικές προτιμήσεις άδειας.

Specific

Ο υπάλληλος δηλώνει την Έναρξη και την Λήξη ημερομηνίας προτίμησης.

Non-Specific

Ο υπάλληλος δηλώνει ένα αριθμό ημερών άδειας που θέλει να λάβει σε ένα χρονικό διάστημα

Min Max

Ο υπάλληλος δηλώνει ένα ελάχιστο και ένα μέγιστο αριθμό ημερών άδειας που θέλει να λάβει σε ένα χρονικό διάστημα.

Παρουσιάζονται εικόνες από την εφαρμογή για καλύτερη κατανόηση.

Στην εικόνα (3.1) βλέπετε τις προτιμήσεις άδειας ενός υπαλλήλου ανά είδος (Bid Type)

Bid Type ▼	Priority Level ▼	Date From ▼	Date To ▼	Number Of D ▼	Min ▼	Max ▼
▼	▼	▼	▼	▼	▼	▼
Min_Max	1	01/01/2024	10/01/2024	0	8	10
Non_Specific	2	01/01/2024	10/01/2024	9	0	0
Specific	3	01/01/2024	10/01/2024	10	0	0

Εικόνα 3.4: Προτιμήσεις άδειας

Στην εικόνα (3.2) βλέπετε τις παραγόμενες άδειες από τις προτιμήσεις της εικόνας (3.1)

Bid Type ▼ ²	Priority Level ▼ ¹	Date From ▼	Date To ▼	Number Of ▼
▼	▼	▼	▼	▼
Bid Type : Min_Max - 6 Items				
Min_Max	1	01/01/2024	10/01/2024	10
Min_Max	1	01/01/2024	09/01/2024	9
Min_Max	1	02/01/2024	10/01/2024	9
Min_Max	1	01/01/2024	08/01/2024	8
Min_Max	1	02/01/2024	09/01/2024	8
Min_Max	1	03/01/2024	10/01/2024	8
Bid Type : Non_Specific - 2 Items				
Non_Specific	2	01/01/2024	09/01/2024	9
Non_Specific	2	02/01/2024	10/01/2024	9
Bid Type : Specific - 1 Items				
Specific	3	01/01/2024	10/01/2024	10

Εικόνα 3.5: Παραγόμενες άδειες ανά προτίμηση

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

Παρατηρούμε ότι η προτίμηση άδειας non-specific με priority level 2 μας παράγει 2 specific άδειες με ημερομηνίες 01/01 έως 09/01 και από 02/01 έως 10/01

Η Πράξη που εκτελείται για την εύρεση του αριθμού παραγόμενων specific από μια προτίμηση non specific είναι η παρακάτω:

$$\text{Number of Specific bids} = (\text{Date to} - \text{Date from}) - \text{Number of Days} + 2$$

Στο δικό μας παράδειγμα η πράξη είναι:

$$\text{Number of Specific bids} = (10 - 1) - 9 + 2 = 2$$

Παρατηρούμε ότι η προτίμηση άδειας min max με priority level 1 μας παράγει 3 non-specific άδειες με αριθμό ημερών 8,9,10. Για να υπολογίσουμε των παραγόμενο αριθμό non-specific μιας προτίμησης min max εκτελούμε την παρακάτω πράξη :

$$\text{Number of Non Specific Bids} = (\text{Max} - \text{Min}) + 1$$

Στο δικό μας παράδειγμα:

$$\text{Number of Non Specific Bids} = (10 - 8) + 1 = 3$$

Οι παραγόμενες non specific άδειες με την σειρά τους παράγουν τις δικές τους specific άδειες.

Επεξήγηση παραγόμενων αδειών από τις προτιμήσεις του υπαλλήλου

Ως Y_{ij} θεωρούμε τις προτιμήσεις άδειας , ως R_{ijr} θεωρούμε τις παραγόμενες non specific και Y_{ijrz} τις παραγόμενες specific.

Ανεξάρτητα τον τύπο προτίμησης άδειας , παράγονται πρώτα non specific άδειες ως R_{ijr} και έπειτα specific άδειες Y_{ijrz} .

Για παράδειγμα:

- Έστω μια Specific άδεια Y_{13} από 01/01 έως 10/01 .Παράγεται μια non specific άδεια R_{131} από 01/01 έως 10/01 με αριθμό ημερών 10 ,αρά στη 'ιδιότητες' πράττει σαν specific.Αυτή με τη σειρά της παράγει μια Y_{1311} specific που είναι ίδια με την προτίμηση.
- Έστω μια non specific προτίμηση Y_{12} από 01/01 έως 10/01 με 9 μέρες. Αυτή παράγει μια non specific R_{121} με τις ίδιες ιδιότητες όπου με την σειρά παράγει την specific Y_{1211} από 01/01 έως 09/01 και την Y_{1212} από 02/01 έως 10/01.
- Έστω μια min max προτίμηση Y_{11} από 01/01 έως 10/01 με min = 8 ημέρες και max = 10 μέρες. Αυτή παράγει $10 - 8 + 1 = 3$ non specific άδειες $R_{111}, R_{112}, R_{113}$ όπου με τη σειρά τους παράγουν τις αντίστοιχες Y_{ijrz} specific άδειες.

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

Στη εικόνα (3.3) βλέπετε την επέκταση της εικόνας (3.2) όπου αναλύεται η σειρά παραγωγής αδειών ανάλογα με το είδος προτίμησης άδειας. Επισυνάπτεται ξανά η (3.1) για καλύτερη κατανόηση.

Bid Type	Priority Level	Date From	Date To	Number Of D	Min	Max
Min_Max	1	01/01/2024	10/01/2024	0	8	10
Non_Specific	2	01/01/2024	10/01/2024	9	0	0
Specific	3	01/01/2024	10/01/2024	10	0	0

Εικόνα 3.6: Προτιμήσεις άδειας

	Bid Type	Priority Level	Date From	Date To	Number Of Days	Yij	Rijr	Yijrz
>	Rijr : R111 - 1 Items							
	Min_Max	1	01/01/2024	10/01/2024	10	Y11	R111	Y1111
	Rijr : R112 - 2 Items							
	Min_Max	1	01/01/2024	09/01/2024	9	Y11	R112	Y1121
	Min_Max	1	02/01/2024	10/01/2024	9	Y11	R112	Y1122
	Rijr : R113 - 3 Items							
	Min_Max	1	01/01/2024	08/01/2024	8	Y11	R113	Y1131
	Min_Max	1	02/01/2024	09/01/2024	8	Y11	R113	Y1132
	Min_Max	1	03/01/2024	10/01/2024	8	Y11	R113	Y1133
	Rijr : R121 - 2 Items							
	Non_Specific	2	01/01/2024	09/01/2024	9	Y12	R121	Y1211
	Non_Specific	2	02/01/2024	10/01/2024	9	Y12	R121	Y1212
	Rijr : R131 - 1 Items							
	Specific	3	01/01/2024	10/01/2024	10	Y13	R131	Y1311

Εικόνα 3.7: Εκτενής ανάλυση παραγόμενων αδειών ανά είδος προτίμησης

Υπολογισμός Ημερομηνίας έναρξης , λήξης παραγόμενης άδειας με βάση του δείκτες r,z

Η ανάλυση θα γίνει με βάση στην non Specific Y_{12} με NDB = αριθμός ημερών πρότασης = 9 και χρονικό ορίζοντα από DF = ημερομηνία έναρξης = 01/01 και ημερομηνία λήξης = 10/01. Η συνάρτηση λειτουργεί με την χρήση των όρων Max και Min .

Για είδος specific, non specific θέτουμε $Max = Min = NDB$

R = Αριθμός παραγόμενων non specific bids ανά j.

Z = Αριθμός παραγόμενων specific bids ανά j, r.

N = Αριθμός ημερών των παραγόμενων specific ανά j, r.

$$R = (Max - Min) + 1 = (9 - 9) + 1 = 1$$

$$N = NDB - r + 1 = 9 - 1 + 1 = 9$$

$$Z = (DT - DF) - N + 2 = (10 - 1) - 9 + 2 = 2$$

Οι ημερομηνίες έναρξης και λήξης των παραγόμενων specific

Ημερομηνία έναρξης = $DF + z - 1$

Ημερομηνία λήξης = $DF + z + NDB - r$

3.3) Μορφοποίηση του προβλήματος

Λείκτες:

$t = 1, 2, 3, \dots, T$ είναι οι μέρες, όπου το T είναι ο χρονικός ορίζοντας
 $I = 1, 2, 3, \dots, I$ είναι οι υπάλληλοι, όπου I is ο αριθμός των υπαλλήλων
 $j = 1, 2, 3, \dots, J$ είναι οι προτάσεις/προσφορά άδειας j του υπαλλήλου i , όπου J είναι ο μέγιστος αριθμός προτάσεων ανά i
 $r = 1, 2, 3, \dots, R$ είναι η non-specific άδεια r της πρότασης j του υπαλλήλου i , όπου R είναι ο μέγιστος αριθμός non-specific αδειών ανά j
 $z = 1, 2, 3, \dots, Z$ είναι η specific άδεια z της non-specific άδειας r της πρότασης j , όπου Z είναι ο μέγιστος αριθμός specific αδειών ανά r

Παράμετροι:

Bmax : Μέγιστος αριθμός ικανοποιημένων προτάσεων άδειας για κάθε υπάλληλο
Se : Τιμή διαχωρισμού μεταξύ 2 ικανοποιημένων προτάσεων
LL_t : Αριθμός υπαλλήλων που επιτρέπεται να βρίσκονται σε άδεια την ημέρα t , $t \in T$
E_i : Ημέρες Άδειας που δικαιούται ο υπάλληλος i , $i \in I$
JB_i : Αριθμός προτάσεων αδειών του υπαλλήλου i , $i \in I$
DF_{ij} : Ημερομηνία έναρξης της πρότασης άδειας j του υπαλλήλου i , $i \in I, j \in J$
DT_{ij} : Ημερομηνία λήξης της πρότασης άδειας j του υπαλλήλου i , $i \in I, j \in J$
NDB_{ij} : Ημέρες άδειας της πρότασης άδειας j του υπαλλήλου i , $i \in I, j \in J$. Αν ο τύπος άδειας είναι Min-Max τότε οι ημέρες άδειας = Μέγιστες ημέρες άδειας.
RB_{ij} : Αριθμός Non-Specific αδειών της πρότασης άδειας j του υπαλλήλου i , $i \in I, j \in J$
ZB_{ijr} : Αριθμός Specific αδειών της non-specific άδειας r της πρότασης άδειας j του υπαλλήλου i , $i \in I, j \in J$

Μεταβλητές Απόφασης:

X_{it} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν στον υπάλληλο i του δοθεί άδεια την ημέρα t , ενναλλακτικά η τιμή είναι 0, $i \in I, t \in T$
R_{ijr} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν στον υπάλληλο i του δοθεί η Non-Specific άδεια r της πρότασης άδειας j , ενναλλακτικά η τιμή είναι 0, $i \in I, j \in J, r \in R$
Y_{ijrz} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν στον υπάλληλο i του δοθεί η Specific άδεια z της Non-Specific άδειας r της πρότασης άδειας j , ενναλλακτικά η τιμή είναι 0, $i \in I, j \in J, r \in R, z \in Z$
Χρησιμοποιώντας τους παραπάνω συμβολισμούς, το πρόβλημα διατυπώνεται ως εξής :

$$\text{s.t. } \sum_{j=1}^{JB_i} \sum_{r=1}^{RB_{ij}} R_{ijr} \leq Bmax, \forall i \in I \quad (2)$$

$$\sum_{j=1}^{JB_i} \sum_{r=1}^{RB_{ij}} R_{ijr} (NDB_{ij} - r + 1) \leq E_i, \forall i \in I \quad (3)$$

$$\sum_{i=1}^I X_{it} \leq LL_t \forall t; \forall t \in T \quad (4)$$

$$Y_{i_1 r_1 z_1} + Y_{i_2 r_2 z_2} \leq 1 \quad \forall i, j_1, r_1, z_1, j_2, r_2, z_2 \text{ if separation or overlap} \quad (5)$$

$$\sum_{t=DF_{ij}+z-1}^{DF_{ij}+z+NDB_{ij}-r} X_{it} \geq (NDB_{ij} - r + 1) Y_{ijrz}, \forall i, j, r, z; \quad (6)$$

$$\sum_{z=1}^{ZB_{ijr}} Y_{ijrz} = R_{ijr} \quad \forall i, j, r; \quad (7)$$

$$X_{it}, R_{ijr}, Y_{ijrz} \text{ binary } \forall i, j, r, z$$

Εικόνα 3.8: Μορφοποίηση Μοντελοποίησης

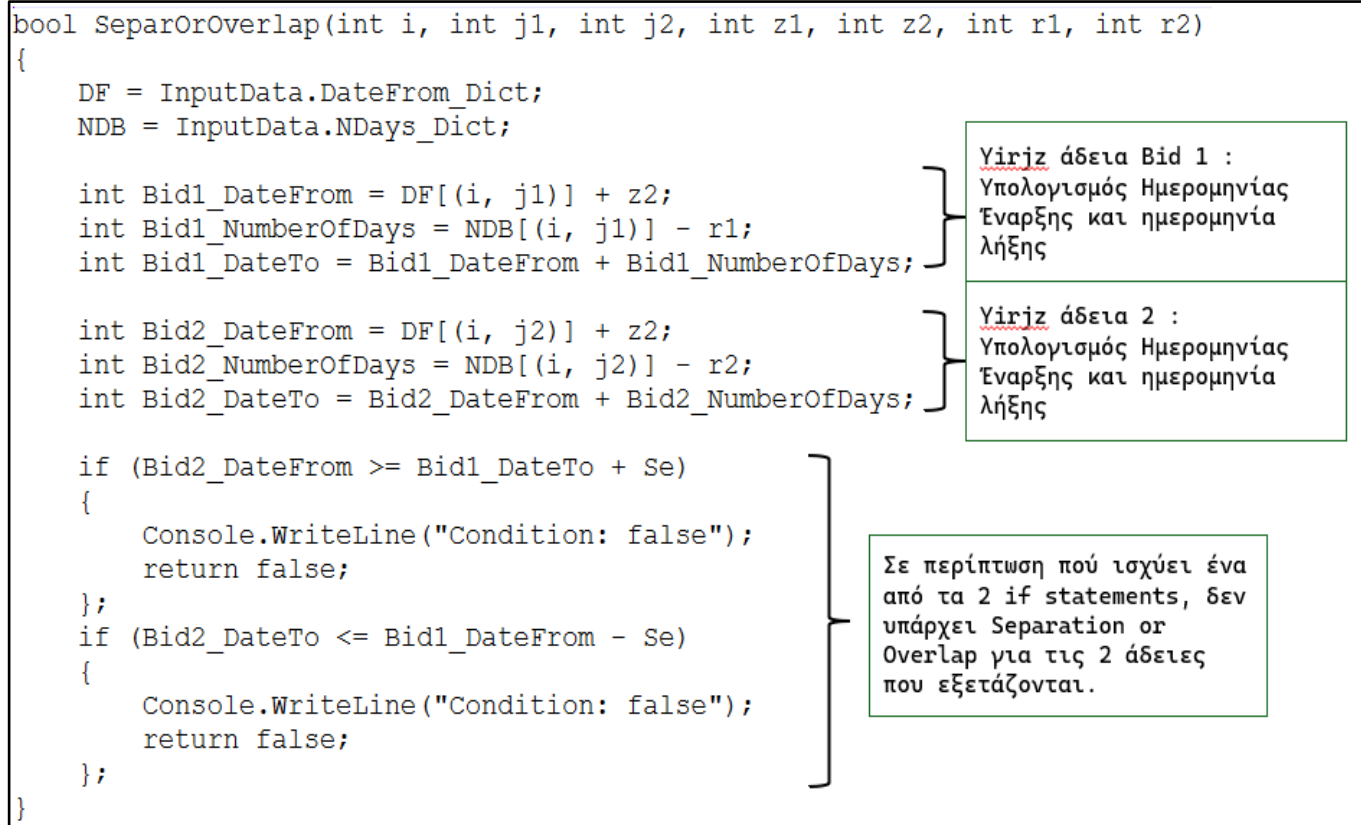
Ο περιορισμός (2) περιορίζει τον αριθμό χορηγημένων παραγόμενων non-specific R_{ijr} να είναι μικρότερος του μέγιστου αριθμού προτιμήσεων $Bmax$ για κάθε υπάλληλο i .

Ο περιορισμός (3) δεν επιτρέπει το συνολικό άθροισμα ημερών $(NDB_{ij} - r + 1)$ των R_{ijr} να ξεπερνά το υπολειπόμενο αριθμό ημερών άδειας E_i του κάθε υπαλλήλου i .

Ο περιορισμός (4) δεν επιτρέπει την χορήγηση άδειας σε περισσότερους υπαλλήλους από το όριο απαιτήσεων LL_t για κάθε ημέρα t .

Ο περιορισμός (5) δεν επιτρέπει να χορηγηθούν 2 άδειες σε ένα υπάλληλο όταν αυτές οι 2 επικαλύπτονται ή το μεταξύ διάστημα ημερών τους να είναι μικρότερο ή ίσο του Se .

Παρατίθεται η Εικόνα (3.5) με κώδικα από την εφαρμογή της εντολής Separation or Overlap :



Εικόνα 3.9: Separation or Overlap

Ο περιορισμός (6) αποσκοπεί στο να συνδεθούν οι μεταβλητές Y_{ijrz} με τις X_{it} για κάθε παραγόμενη specific άδεια .

Ανάλυση των ορίων του αθροίσματος X_{it} .

Κάτω όριο: Ημερομηνία έναρξης άδειας = $DF_{ij} + z - 1$.

Αριθμός ημερών άδειας = $NDB_{ij} - r + 1$.

Άνω όριο: Ημερομηνία λήξης άδειας = $DF_{ij} + z - 1 + \text{Αριθμός ημερών} = DF_{ij} + z + NDB_{ij} - r$.

Ο περιορισμός (7) αποσκοπεί στη σύνδεση των μεταβλητών R_{ijr} , Y_{ijrz} . Όταν η τιμή της μεταβλητής R_{ijr} είναι ίση με 1, μόνο 1 ένα παραγόμενο Y_{ijrz} να λαμβάνει την τιμή 1. Σε περίπτωση που η τιμή της R_{ijr} είναι ίση με 0, καμία παραγόμενη specific άδεια Y_{ijrz} δεν μπορεί να λάβει την τιμή 1.

Εέλεγχος χορήγησης προτιμήσεων αδειών .

Με βάση τη λογική ανάθεσης (Strict Seniority, Fair Priority, Bib by Bid) θέτουμε το κατώτερο όριο (Lower της εκάστοτε επιλεγμένης R_{ijr} ως 1. Ελέγχουμε αν μπορεί να ικανοποιηθεί. Αν ναι κλειδώνουμε την τιμή του κατώτερου ορίου και συνεχίζουμε έλεγχο των επόμενων R_{ijr} . Σε περίπτωση μη χορήγησης επαναφέρουμε το κατώτερο όριο ως 0.

Να σημειωθεί ότι με την χορήγηση μιας R_{ijr} δεν κλειδώνουμε μια συγκεκριμένη Y_{ijrz} των παραγόμενων specific Y_{ijrz} της R_{ijr} . Συνεπώς η τελική άδεια Y_{ijrz} με τις ακριβής ημερομηνίες που θα χορηγηθεί τις δύναται ελεύθερο το περιθώριο να μεταβάλλεται συνεχώς κατά την διάρκεια ελέγχου των υπολειπόμενων μη ελεγμένων αδειών.

Να υπενθυμίσουμε σε αυτό το σημείο ότι η non-specific R_{ijr} που παράγονται από τις προτιμήσεις αδειών του κάθε υπαλλήλου είναι 1 προς 1, εκτός από την περίπτωση του τύπου άδειας = Min Max όπου ο αριθμός των παραγόμενων R_{ijr} είναι ίσος με Max - Min + 1. Σε περίπτωση που χορηγείται η επιλεγμένη R_{ijr} με τύπο Min Max κάνουμε break από την επανάληψη. Έτσι καταφέρνουμε να χορηγήσουμε τον μεγαλύτερο αριθμό ημερών που μπορούμε σε μια Min Max.

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

Παρατίθεται εικόνα (3.6) του κώδικα για καλύτερη κατανόηση της τεχνικής:

Θέσε το άνω, κάτω όριο της επιλεγμένης R_{ijr} = 1;

Έλεγχος χορήγησης επιλεγμένης R_{ijr}. Σε περίπτωση χορήγησης, θέσε το Rvalue = 0

Σε περίπτωση απόρριψης θέσε άνω, κάτω όρια = 0.

```
for (r = 0; r < Rvalue; r++)
{
    R[EmpId, j, r].LB = 1;
    R[EmpId, j, r].UB = 1;

    model.Solve();
    if (model.GetStatus() == Cplex.Status.Optimal)
    {
        string message = $"Crew member {EmpId + 1} was awarded bid {j + 1}";
        Console.WriteLine(message);
        grant = true;

        //Update Latest Feasible Model ,ObjectiveValue
        OutputData.ObjValue = model.ObjValue;
        FeasibleModel = model;

        //We set Rvalue = 0 in order to break from the R loop
        Rvalue = 0;
    }
    else
    {
        grant = false;

        R[EmpId, j, r].LB = 0;
        R[EmpId, j, r].UB = 0;

        string message = $"Crew member {EmpId + 1} was not awarded bid {j + 1}";
        Console.WriteLine(message);
    }
}
```

Εικόνα 3.10: Εικόνα Κώδικα Ελέγχου R_{ijr}

Τέλος, παρατίθεται κώδικας για κατανόηση των λογικών ανάθεσης αδειών (Strict Seniority, Fair Priority, Bid by Bid).

Ανάλυση μεταβλητών
Empid = δείκτης i του υπάλληλου.
Grant = true ,αν χορηγήθηκε το προηγούμενο bid του empid
Rid = Επόμενος υπάλληλος
Logic = Λογική ανάθεσης

```
if (logic == BasicEnums.VPLogicType.Strict_Seniority)
{
    Rid = empId;
}
else if (logic == BasicEnums.VPLogicType.Fair_Assignment)
{
    if (grant == true)
    {
        Rid = empId + 1;
    }
    else
    {
        Rid = empId;
    }
}
else if (logic == BasicEnums.VPLogicType.Bid_By_Bid)
{
    Rid = empId + 1;
}
```

Strict Seniority
Ανεξαρτήτως της τιμής grant του προηγούμενου Bid που ελέγχθηκε, συνέχισε με τον ίδιο υπάλληλο.

Fair Priority
Αν grant = true επέλεξε τον επόμενο υπάλληλο στην ιεραρχία. Αν grant = false συνέχισε με τον ίδιο υπάλληλο.

Strict Seniority
Ανεξαρτήτως της τιμής grant του προηγούμενου Bid που ελέγχθηκε, συνέχισε με τον επόμενο υπάλληλο

Εικόνα 3.11: Εικόνα Κώδικα Λογικής Ανάθεσης

3.4) Κώδικας βελτιστοποίησης σε C#, IBM CPLEX

Παρατίθεται η μορφοποίηση σε c# και cplex .

3.4.1) Δείκτες, παράμετροι.

```
#region Indexes
```

```
int I = InputData.Employees.Count();
int T = InputData.Dates.Count();
int Jmax = InputData.N_Dict.Max(kvp => kvp.Value);
int Rmax = InputData.RBids_Dict.Max(kvp => kvp.Value);
int Zmax = InputData.ZBids_Dict.Max(kvp => kvp.Value);
```

```
#endregion
```

```
#region Parameters
```

```
int Bmax = InputData.Bmax;
int Se = InputData.Se;
```

```
Dictionary<int, int> LL = InputData.LLi_Dict;
Dictionary<int, int> E = InputData.MaxD_Dict;
Dictionary<(int, int), int> DF = InputData.DateFrom_Dict;
Dictionary<(int, int), int> DT = InputData.DateTo_Dict;
Dictionary<(int, int), int> NDB = InputData.NDays_Dict;
Dictionary<int, int> JB = InputData.N_Dict;
Dictionary<(int, int), int> RB = InputData.RBids_Dict;
Dictionary<(int, int, int), int> ZB = InputData.ZBids_Dict;
```

```
#endregion
```

```
#region Model Initialization
```

```
Cplex model = new Cplex();
Cplex FeasableModel = new Cplex();
```

```
#endregion
```

3.4.2) Μεταβλητές απόφασης , Αντικειμενική συνάρτηση.

```
#region Decision Variables
```

```
INumVar[, ,] Y = new INumVar[I, Jmax, Rmax, Zmax];
INumVar[,] X = new INumVar[I, T];
INumVar[,] R = new INumVar[I, Jmax, Rmax];
```

```
// Create decision variables X
for (int i = 0; i < I; i++)
    for (int t = 0; t < T; t++)
    {
        string varNameX = $"X{i + 1}_{t + 1}";
        X[i, t] = model.NumVar(0, 1, NumVarType.Bool, varNameX);
    }
```

```
// Create decision variables Y
for (int i = 0; i < I; i++)
    for (int j = 0; j < JB[i]; j++)
```

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

```
for (int r = 0; r < RB[(i, j)]; r++)
    for (int z = 0; z < ZB[(i, j, r)]; z++)
    {
        string varNameY = $"Y{i + 1}_{j + 1}_{r + 1}_{z + 1}";

        Y[i, j, r, z] = model.NumVar(0, 1, NumVarType.Bool, varNameY);
    }

// Create decision variables R
for (int i = 0; i < I; i++)
    for (int j = 0; j < JB[i]; j++)
        for (int r = 0; r < RB[(i, j)]; r++)
        {
            string varNameR = $"R{i + 1}_{j + 1}_{r + 1}";

            R[i, j, r] = model.NumVar(0, 0, NumVarType.Bool, varNameR);
        }
#endregion

#region Objective Function

ILinearNumExpr objective = model.LinearNumExpr();

for (int i = I - 1; i >= 0; i--)
    for (int j = JB[i] - 1; j >= 0; j--)
        for (int r = 0; r < RB[(i, j)]; r++) //allagh
        {
            objective.AddTerm(1.0, R[i, j, r]);
        }

model.AddMaximize(objective);

#endregion
```

3.4.3) Περιορισμοί.

```
#region Constrains

// #2. Maximum Satisfied Bids Per Employee
for (int i = 0; i < I; i++)
{
    ILinearNumExpr sumLeaveBids = model.LinearNumExpr();

    for (int j = 0; j < JB[i]; j++)
    {
        for (int r = 0; r < RB[(i, j)]; r++)
        {
            sumLeaveBids.AddTerm(1.0, R[i, j, r]);
        }
    }

    model.AddLe(sumLeaveBids, Bmax, "Con2_" + (i+1));
}

// #3. Entitlements
for (int i = 0; i < I; i++)
{
```

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

```
ILinearNumExpr SumNDB = model.LinearNumExpr();
for (int j = 0; j < JB[i]; j++)
{
    var NDays = NDB[(i,j)];

    for (int r = 0; r < RB[(i, j)]; r++)
    {
        NDays = NDB[(i,j)] - r;

        SumNDB.AddTerm(NDays, R[i, j, r]);
    }
}
model.AddLe(SumNDB, E[i], "Con3_" + (i + 1));
}
// #4. Limit Lines
for (int t = 0; t < T; t++)
{
    ILinearNumExpr expr = model.LinearNumExpr();

    int sumdays = 0;

    for (int i = 0; i < I; i++)
    {
        expr.AddTerm(1, X[i, t]);
        sumdays++;
    }

    if (sumdays > LL[t])
    {
        model.AddLe(expr, LL[t], "Con4_" + (t + 1));
    }
}

//#5. Separation or Overlapping

for (int i = 0; i < I; i++)
    for (int j1 = 0; j1 < JB[i] - 1; j1++)
        for (int j2 = j1 + 1; j2 < JB[i]; j2++)
            for (int r1 = 0; r1 < RB[(i, j1)]; r1++)
                for (int r2 = 0; r2 < RB[(i, j2)]; r2++)
                    for (int z1 = 0; z1 < ZB[(i, j1, r1)]; z1++)
                        for (int z2 = 0; z2 < ZB[(i, j2, r2)]; z2++)
                            if (SeparOrOverlap(i, j1, j2, z1, z2, r1, r2))
                                {
                                    ILinearNumExpr expr = model.LinearNumExpr();
                                    expr.AddTerm(1, Y[i, j1, r1, z1]);
                                    expr.AddTerm(1, Y[i, j2, r2, z2]);
                                    model.AddLe(expr, 1, $"Con5_{i + 1}_{j1 + 1}_{z1 + 1}_{j2 + 1}_{z2 + 1}");
                                }

bool SeparOrOverlap(int i, int j1, int j2, int z1, int z2, int r1, int r2)
{
    int Bid1_DateFrom = InputData.DateFrom_Dict[(i, j1)];
    int Bid2_DateFrom = InputData.DateFrom_Dict[(i, j2)];
    int Bid1_NBDays = InputData.NDays_Dict[(i, j1)];
    int Bid2_NBDays = InputData.NDays_Dict[(i, j2)];

    if (Bid2_DateFrom + z2 >= Bid1_DateFrom + Bid1_NBDays + Se + z1 - r1 )
```

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

```
{
    Console.WriteLine("Condition: false");
    return false;
};

if (Bid2_DateFrom + Bid2_NBDays + z2 -r1 <= Bid1_DateFrom - Se + z1)
{
    Console.WriteLine("Condition: false");
    return false;
};

Console.WriteLine("Condition: True");
return true;
}

//#6. Connection Between Y and X
for (int i = 0; i < I; i++)
    for (int j = 0; j < JB[i]; j++)
    {
        int NumberOfDays = NDB[(i, j)];

        for (int r = 0; r < RB[(i, j)]; r++)
        {
            NumberOfDays = NumberOfDays - r;
            for (int z = 0; z < ZB[(i, j, r)]; z++)
            {
                ILinearNumExpr expr = model.LinearNumExpr();
                expr.AddTerm(NumberOfDays, Y[i, j, r, z]);

                int start = DF[(i, j)] + z;
                int end = start + NumberOfDays;

                for (int t = start; t < end; t++)
                {
                    expr.AddTerm(-1, X[i, t]); // Add X variables for each day of the
bid
                }
                model.AddLe(expr, 0, $"Con6_{i + 1}_{j + 1}_{r + 1}_{z + 1}");
            }
        }
    }

//#7. Connection Between Y and R
for (int i = 0; i < I; i++)
    for (int j = 0; j < JB[i]; j++)
        for (int r = 0; r < RB[(i, j)]; r++)
        {
            ILinearNumExpr sumYijrz = model.LinearNumExpr();
            for (int z = 0; z < ZB[(i, j, r)]; z++)
            {
                sumYijrz.AddTerm(1.0, Y[i, j, r, z]);
            }

            model.AddEq(R[i, j, r], sumYijrz, $"Con7_{i + 1}_{j + 1}_{r + 1}");
        }
#endregion
```

3. ΑΝΑΘΕΣΗ ΑΔΕΙΩΝ ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΠΡΟΤΙΜΗΣΕΙΣ ΤΩΝ ΥΠΑΛΛΗΛΩΝ

3.4.4) Έλεγχε προτίμηση άδειας με βάση τη λογική ανάθεσης

```
#region Check Bids (Strict Seniority, Fair Priority, Bid by Bid)

//Bids Assignment Logic e.x Strict Seniority, Fair Priority , Bid by Bid
BasicEnums.VPLogicType logic = InputData.VPLogicType;

//Total Number of Bids
int N = JB.Values.Sum();

int[] NextBid = new int[N];

//Transfom N_Dict to Array
int[] NrOfBids = JB.Values.ToArray();

//Μετρητες για τα ολοκληρωμένα Bids και Employees
int FinishedBids = 1;
int FinishedEmpIds = 0;

//Support Variables Initialize
bool grant = false;
int EmpId = 0;

while (FinishedEmpIds <= I)
{
    int j = NextBid[EmpId];

    #region Select Rijr

    var r = 0;
    int Rvalue = new int();
    Rvalue = RB[(EmpId, j)];

    #endregion
    for (r = 0; r < Rvalue; r++)
    {
        #region Check Bid

        R[EmpId, j, r].LB = 1;
        R[EmpId, j, r].UB = 1;

        model.Solve();
        if (model.GetStatus() == Cplex.Status.Optimal)
        {
            Console.WriteLine(message);
            grant = true;

            //Update Latest Feasable Model ,ObjectiveValue
            OutputData.ObjValue = model.ObjValue;
            FeasableModel = model;

            //We set Rvalue = 0 in order to break from the R Loop
            Rvalue = 0;
        }
        else
        {
            grant = false;
            R[EmpId, j, r].LB = 0;
            R[EmpId, j, r].UB = 0;
        }
    }
}
```

```
        #endregion
    }

    NextBid[EmpId]++;
    if (NextBid[EmpId] == NrOfBids[EmpId])
    {
        FinishedEmpIds++;
    }
    if (FinishedBids == N)
    {
        break;
    }
    FinishedBids = FinishedBids + 1;
    if (FinishedBids <= N)
    {
        EmpId = GetNextId(EmpId, grant, I, NextBid, NrOfBids, FinishedEmpIds,
logic);
    }
}
public int GetNextId(int aId, bool accept, int N, int[] NextBid,
    int[] NrOfBids,int FinishedEmpIds,
        BasicEnums.VPLogicType VPLogicType)
{
    try
    {
        int RId = 0;
        var logic = VPLogicType;
        if (FinishedEmpIds == N)
            return 0;

        if (logic == BasicEnums.VPLogicType.Strict_Seniority)
        {
            RId = aId;
        }
        else if (logic == BasicEnums.VPLogicType.Fair_Assignment)
        {
            if (accept == true)
            {
                RId = aId + 1;
            }
            else
            {
                RId = aId;
            }
        }
        else if (logic == BasicEnums.VPLogicType.Bid_By_Bid)
        {
            RId = aId + 1;
        }

        if (RId == N)
        {
            RId = 0;
        }
        while (NextBid[RId] == NrOfBids[RId])
        {
            RId++;
            if (RId == N)
            {
                RId = 0;
            }
        }
    }
}
```

```
        }
    }

    return RId;
}
catch (System.Exception ex)
{
    Console.WriteLine("An error occurred: " + ex.Message);
    return -1;
}
}
```

3.4.5) Εξαγωγή Αποτελεσμάτων

```
if (OutputData.ObjValue > 0.1)
{
    FeasableModel.Solve();
    using (FeasableModel)
    {
        FeasableModel.Solve();

        #region Extract CPLEX Model, Solution

        #region Filepath

        string relativePath = Path.Combine("OptimizationResults", "CPLEX",
                                            "Thesis", "Vacation_Planning");

        #endregion

        FeasableModel.ExportModel(Path.Combine(relativePath, "VP.lp"));
        FeasableModel.ExportModel(Path.Combine(relativePath, "VP.mps"));
        FeasableModel.WriteSolution(Path.Combine(relativePath, "VP.sol"));
        OutputData.ObjValue = FeasableModel.ObjValue;

        #endregion
    }
}
```

Στο κεφάλαιο 4 αναλύεται το πρόβλημα της αρχικοποίησης του Column Generation.

- Στο 4.1 παρουσιάζεται μια σύντομη ανάλυση του προβλήματος.
- Στο 4.2 παρουσιάζεται η μορφοποίηση του προβλήματος.
- Στο 4.3 παρουσιάζεται ο κώδικας.
- Στο 4.4 παρουσιάζεται ένα υπολογιστικό παράδειγμα με τα παραγόμενα .sol , .lp

4.1) Ανάλυση του προβλήματος

Το πρόβλημα αφορά την ανάθεση ημερών άδειας σε κάθε μέλος του προσωπικού που δεν έχουν ληφθεί από το πρόβλημα ανάθεσης αδειών που αναλύθηκε στο κεφάλαιο 3 .Η ανάθεση γίνεται τυχαία χωρίς να λαμβάνονται υπόψη οι προτιμήσεις των υπαλλήλων. Στο τέλος, η εταιρία θα έχει κόστος για κάθε υπάλληλο που δεν έχει εξαντλήσει όλες τις ημέρες άδειάς του. Συνεπώς, ο μοναδικός μας στόχος είναι η πλήρης εξάλειψη των υπολοίπων ημερών άδειας για κάθε μέλος του προσωπικού.

Εμείς προχωρούμε μόνο με την αρχικοποίηση του προβλήματος.

4.2) Μορφοποίηση του προβλήματος

Δείκτες:

$t = 1, 2, 3, \dots, T$ είναι οι ημέρες, όπου T είναι ο χρονικός ορίζοντας.

$i = 1, 2, 3, \dots, I$ είναι οι υπάλληλοι, όπου I is ο αριθμός των υπαλλήλων.

$j = 1, 2, 3, \dots, R_i$ είναι τα πλανά άδειας, όπου R_i είναι το σύνολο των εναλλακτικών πλάνων του υπαλλήλου i .

Παράμετροι:

c_{ij} : Κόστος του πλάνου άδειας j του υπαλλήλου i ,

a_{ijt} : Δυαδική παράμετρος που παίρνει την τιμή 1 αν το πλάνο j του υπαλλήλου i καλύπτει την ημέρα t , εναλλακτικά η τιμή είναι 0, $i \in I, t \in T, j \in R_i$

Μεταβλητές Απόφασης:

w_{ij} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν το πλάνο άδειας j του υπαλλήλου i ανατίθεται στον υπάλληλο, εναλλακτικά η τιμή είναι 0, $i \in I, j \in R_i$

Χρησιμοποιώντας τους παραπάνω συμβολισμούς, το πρόβλημα διατυπώνεται ως εξής :

$$\text{Min } \sum_{i \in I} \sum_{j \in R_i} c_{ij} w_{ij} \quad (8)$$

$$\text{s.t. } \sum_{j \in R_i} w_{ij} = 1, \forall i \in I \quad (9)$$

$$\sum_{i \in I} \sum_{j \in R_i} a_{ijt} w_{ij} \geq 0, \forall t \in T \quad (10)$$

$$\sum_{i \in I} \sum_{j \in R_i} a_{ijt} w_{ij} \leq RLL_t, \forall t \in T \quad (11)$$

$$w_{ij} \text{ binary}, \forall i, t, j \in R_i$$

Η αντικειμενική συνάρτηση (8) ελαχιστοποιεί το συνολικό κόστος, το οποίο είναι άμεσα ανάλογο με το συνολικό αριθμό των μη χορηγημένων ημερών διακοπών.

Ο περιορισμός (9) εξασφαλίζει ότι σε κάθε υπάλληλο θα του εκχωρηθεί ένα μόνο πλάνο διακοπών.

Ο περιορισμός (10) χρησιμοποιείται αποκλειστικά για την παρακολούθηση των ημερών που περιλαμβάνει κάθε πλάνο διακοπών.

Ο περιορισμός (11) χρησιμοποιείται για την παρακολούθηση των ημερών διακοπών που περιλαμβάνονται σε κάθε πλάνο, καθώς και για την εξασφάλιση ότι τα εναπομείναντα όρια δεν παραβιάζονται.

4.3) Κώδικας βελτιστοποίησης σε C#, IBM CPLEX.

Παρατίθεται η μορφοποίηση της αρχικοποίησης με υπολογιστικό παράδειγμα .

4.3.1) Αρχικοποίηση δεικτών, παραμέτρων

#region Ri,Cij,Aijt Initialization

```
CGInputdata.I = InputData.Employees.Count();
for (int i = 0; i < CGInputdata.I; i++)
{
    //Create a List with the Empty Vacation Plan
    List<int> Vacationplan_List = new List<int>();
    Vacationplan_List.Add(0);

    //Insert into Ri
    CGInputdata.Ri_Dict.Add(i, Vacationplan_List);

    //Add Cij with Cost = RE[i]
    CGInputdata.Cij_Dict.Add((i, 0), CGInputdata.Re_Dict[i]);
}

CGInputdata.T = InputData.Dates.Count();
for (int t = 0; t < CGInputdata.T; t++)
{
    for (int i = 0; i < CGInputdata.I; i++)
    {
        //For each i,t and j=0 set Ajt = 0
        CGInputdata.Aijt_Dict[(i, 0, t)] = 0;
    }
}

#endregion
```

4.3.2) Δείκτες, παράμετροι

#region Indexes

```
int I = InputData.I;
int T = InputData.T;
int J = InputData.Ri_Dict.Values.Max(list => list.Count);

#endregion
```

#region Parameters

```
//Dictionary<(Employee,List<VacationPlans>), 0 or 1>
Dictionary<int, List<int>> Ri = InputData.Ri_Dict;

//Dictionary<Employee, Residual Limit Line>
Dictionary<int, int> RLLt = InputData.RLLt_Dict;

//Dictionary<(Employee,Vacation Plan), Residual Entitlement Cost>
Dictionary<(int, int), double> Cij = InputData.Cij_Dict;

//Dictionary<(Employee,Vacation Plan Day), 0 or 1>
```

```
Dictionary<(int, int, int), int> Aijt = InputData.Aijt_Dict;
```

```
#endregion
```

4.3.3) Μεταβλητές Απόφασης, αντικειμενική συνάρτηση

```
#region Model Initialization
```

```
Cplex model = new Cplex();
```

```
#endregion
```

```
#region Decision Variables
```

```
INumVar[,] W = new INumVar[I, J];
```

```
for (int i = 0; i < I; i++)
```

```
{
```

```
    List<int> VacationPlans_List = Ri.TryGetValue(i, out var productList) ?  
    productList : new List<int>();
```

```
    foreach (int j in VacationPlans_List)
```

```
    {
```

```
        string varNameX = $"W{i + 1}_{j + 1}";
```

```
        W[i, j] = model.NumVar(0, 1, NumVarType.Bool, varNameX);
```

```
    }
```

```
}
```

```
#endregion
```

```
#region Objective Function
```

```
ILinearNumExpr objective = model.LinearNumExpr();
```

```
for (int i = 0; i < I; i++)
```

```
{
```

```
    List<int> VacationPlans_List = Ri.TryGetValue(i, out var productList) ?  
    productList : new List<int>();
```

```
    foreach (int j in VacationPlans_List)
```

```
    {
```

```
        var Cij_Cost = Cij[(i, j)] * 1000;
```

```
        objective.AddTerm(Cij_Cost, W[i, j]);
```

```
    }
```

```
}
```

```
model.AddMinimize(objective);
```

```
#endregion
```

4.3.4) Περιορισμοί

#region Constraints

```
//#8. Con 8
for (int i = 0; i < I; i++)
{
    ILinearNumExpr expr = model.LinearNumExpr();

    List<int> VacationPlans_List = Ri.TryGetValue(i, out var productList) ?
productList : new List<int>();
    foreach (int j in VacationPlans_List)
    {
        expr.AddTerm(1, W[i, j]);
    }
    model.AddEq(expr, 1, "CON8_" + (i + 1));
}

//#9. Con 9
for (int t = 0; t < T; t++)
{
    ILinearNumExpr expr = model.LinearNumExpr();

    for (int i = 0; i < I; i++)
    {
        List<int> VacationPlans_List = Ri.TryGetValue(i, out var productList)
? productList :
new List<int>();
        foreach (int j in VacationPlans_List)
        {
            expr.AddTerm(Aijt[(i, j, t)], W[i, j]);
        }
    }
    model.Ge(expr, 0, "CON9_" + (t + 1));
}

//#10. Con 10
for (int t = 0; t < T; t++)
{
    ILinearNumExpr expr = model.LinearNumExpr();

    for (int i = 0; i < I; i++)
    {
        List<int> VacationPlans_List = Ri.TryGetValue(i, out var productList)
? productList :
new List<int>();
        foreach (int j in VacationPlans_List)
        {
            expr.AddTerm(Aijt[(i, j, t)], W[i, j]);
        }
    }
    model.AddLe(expr, RLLt[t], "CON10_" + (t + 1));
}

#endregion
```

4.3.5) Εξαγωγή αποτελεσμάτων

```
#region Get Results
model.Solve();
// Solve the model
if (model.GetStatus() == Cplex.Status.Optimal)
{
    #region FilePath

    string relativePath = Path.Combine("OptimizationResults", "CPLEX", "Thesis",
"VP_Column_Generation");
    Directory.CreateDirectory(relativePath);

    #endregion

    Data.ObjValue = model.ObjValue;

    #region Export Model To File

    model.ExportModel(Path.Combine(relativePath, "VP_CG.lp"));
    model.ExportModel(Path.Combine(relativePath, "VP_CG.mps"));
    model.WriteSolution(Path.Combine(relativePath, "VP_CG.sol"));

    #endregion

}
model.End();
#endregion
```

4.4) Υπολογιστικό Παράδειγμα

4.4.1) Περιγραφή δεδομένων εισαγωγής.

Δεδομένα Εισαγωγής :

- T = Χρονικός Ορίζοντας = 10 μέρες
- RLL = Υπολειπόμενο Limit Line ανά ημέρα= 0 ή 1 .
- I =Αριθμός υπαλλήλων = 3
- RE = Υπολειπόμενες μέρες άδειας:

i	Re
1	12
2	9
3	20

4.4.2) Αποτελέσματα: Αρχείο .lp \ENCODING=ISO-8859-1 \Problem name: ILOG.CPLEX Minimize obj1: 12000 W1_1 + 9000 W2_1 + 20000 W3_1 Subject To CON8_1: W1_1 = 1 CON8_2: W2_1 = 1 CON8_3: W3_1 = 1 CON10_1: <= 0 CON10_2: <= 0 CON10_3: <= 0 CON10_4: <= 0 CON10_5: <= 0 CON10_6: <= 0	4.4.3) Αποτελέσματα: Αρχείο .sol <CPLEXSolution version="1.2"> ... <variables> <variable name="W1_1" index="0" value="1"/> <variable name="W2_1" index="1" value="1"/> <variable name="W3_1" index="2" value="1"/> </variables> <objectiveValues> <objective index="0" name="" value="41000"/> </objectiveValues> </CPLEXSolution>
--	--

4. ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΥΧΑΙΑΣ ΑΝΑΘΕΣΗΣ ΥΠΟΛΕΙΠΟΜΕΝΩΝ ΗΜΕΡΩΝ ΑΔΕΙΑΣ

CON10_7: <= 0 CON10_8: <= 0 CON10_9: <= 0 CON10_10: <= 0 Bounds 0 <= W1_1 <= 1 0 <= W2_1 <= 1 0 <= W3_1 <= 1 Binaries W1_1 W2_1 W3_1 End	
--	--

5. CREW ROSTERING: ΑΡΧΙΚΟΠΟΙΗΣΗ SET-COVER, SET-PARTITION

Στο κεφάλαιο 5 αναλύεται το πρόβλημα κατανομής πληρώματος .

Στο 5.1 παρουσιάζονται οι 2 μορφοποιήσεις Set-Cover , Set-Partition,

Στο 5.2.1 παρουσιάζεται το Set-Partition μαζί με υπολογιστικό παράδειγμα .

Στο 5.2.2 παρουσιάζεται το Set-Cover μαζί με υπολογιστικό παράδειγμα .

5.1) Μορφοποίηση Set-Cover, Set-Partition

5.1.1) Set Partition

Δείκτες:

$i = 1, 2, 3, \dots, I$ είναι οι υπάλληλοι , όπου I is ο αριθμός των υπαλλήλων

$f = 1, 2, 3, \dots, F$ είναι τα flight routes , όπου F είναι ο συνολικός αριθμός των Flight Routes

$j = 1, 2, 3, \dots, R_i$ είναι τα rosters του υπαλλήλου i , όπου R_i είναι το σύνολο των rosters του υπαλλήλου i

Παράμετροι:

h : κόστος για κάθε flight route που παραμένει ακάλυπτη ,

b_f : Συνολικός αριθμός υπαλλήλων που χρειάζονται (complement) στην flight route $f, f \in F$

c_{ij} : Κόστος του roster j του υπαλλήλου i ,

a_{ijf} : Δυαδική παράμετρος που παίρνει την τιμή 1 το roster j του υπαλλήλου i καλύπτει την flight route f ,
ενναλλακτικά η τιμή είναι 0, $i \in I, f \in F, j \in R_i$

Μεταβλητές Απόφασης:

X_{ij} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν το roster j του υπαλλήλου i ανατίθεται στον υπάλληλο , ενναλλακτικά η τιμή είναι 0, $i \in I, j \in R_i$

Y_f : Ακαριαία μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν η flight route f παραμείνει ακάλυπτη,
ενναλλακτικά η τιμή είναι 0, $f \in F$.

Χρησιμοποιώντας τους παραπάνω συμβολισμούς , το πρόβλημα διατυπώνεται ως εξής :

$$\text{Min } \sum_{i \in I} \sum_{j \in R_i} c_{ij} X_{ij} + \sum_{f \in F} h Y_f \quad (12)$$

$$\text{s.t. } \sum_{j \in R_i} X_{ij} = 1, \forall i \in I \quad (13)$$

$$Y_f + \sum_{i \in I} \sum_{j \in R_i} a_{ijf} X_{ij} = b_f, \forall f \in F \quad (14)$$

$$X_{ij} \text{ binary}, Y_f \text{ integer}, \forall i, f, j \in R_i$$

Η αντικειμενική συνάρτηση (12) ελαχιστοποιεί το συνολικό κόστος, το οποίο περιλαμβάνει το κόστος των μη καλυμμένων διαδρομών / Flight Routes συν την ποιότητα του προγράμματος.

Ο περιορισμός (13) εξασφαλίζει ότι ακριβώς ένα πρόγραμμα / roster ανατίθεται σε κάθε μέλος του πληρώματος.

Ο περιορισμός (14) δηλώνει ότι κάθε διαδρομή πτήσης πρέπει να καλύπτεται ακριβώς από τον απαιτούμενο αριθμό μελών του πληρώματος.

Η μεταβλητή Y_f γίνεται θετική όταν η διαδρομή πτήσης f δεν καλύπτεται πλήρως, οπότε επιβάλλεται η αντίστοιχη ποινή h στην αντικειμενική συνάρτηση.

5.1.2) Set Cover

Δείκτες:

$i = 1, 2, 3, \dots, I$ είναι οι υπάλληλοι, όπου I is ο αριθμός των υπαλλήλων

$f = 1, 2, 3, \dots, F$ είναι τα flight routes, όπου F είναι ο συνολικός αριθμός των Flight Routes

$j = 1, 2, 3, \dots, R_i$ είναι τα rosters του υπαλλήλου i , όπου R_i είναι το σύνολο των rosters του υπαλλήλου i

Παράμετροι:

h : κόστος για κάθε flight route που παραμένει ακάλυπτη,

b_f : Συνολικός αριθμός υπαλλήλων που χρειάζονται (complement) στην flight route $f, f \in F$

a_{ijf} : Δυαδική παράμετρος που παίρνει την τιμή 1 το roster j του υπαλλήλου i καλύπτει την flight route f , ενναλακτικά η τιμή είναι 0, $i \in I, f \in F, j \in R_i$

Μεταβλητές Απόφασης:

X_{ij} : Δυαδική μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν το roster j του υπαλλήλου i ανατίθεται στον υπάλληλο, ενναλακτικά η τιμή είναι 0, $i \in I, j \in R_i$

Y_f : Ακαριαία μεταβλητή απόφασης όπου παίρνει την τιμή 1 αν η flight route f παραμένει ακάλυπτη, ενναλακτικά η τιμή είναι 0, $f \in F$.

Χρησιμοποιώντας τους παραπάνω συμβολισμούς, το πρόβλημα διατυπώνεται ως εξής :

$$\text{Min } \sum_{f \in F} h Y_f \quad (15)$$

$$\text{s.t. } \sum_{j \in R_i} X_{ij} = 1, \forall i \in I \quad (16)$$

$$Y_f + \sum_{i \in I} \sum_{j \in R_i} a_{ijf} X_{ij} \geq b_f, \forall f \in F \quad (17)$$

$$X_{ij} \text{ binary, } Y_f \text{ integer, } \forall i, f, j \in R_i$$

Υπάρχουν δύο διαφορές μεταξύ αυτής του set-cover και της set-partition διατύπωσης.

Η πρώτη βρίσκεται στην αντικειμενική συνάρτηση (15). Στη set-cover περιλαμβάνεται μόνο το κόστος κάλυψης, ενώ το κόστος ποιότητας του roster παραλείπεται.

Η δεύτερη διαφορά είναι στο σύνολο περιορισμών (17). Αντί για ισότητα, εδώ χρησιμοποιούμε περιορισμούς ανισότητας ($\geq$). Αυτό το σύνολο περιορισμών αφορά την κάλυψη των διαδρομών (Routes). Στη set-partition, αυτοί οι περιορισμοί δηλώνουν ότι ο αριθμός των μελών του πληρώματος που καλύπτουν κάθε pairing πρέπει να είναι ίσος με το b_f . Στη set-cover, το άθροισμα των μελών του πληρώματος που καλύπτουν κάθε flight route μπορεί να είναι μεγαλύτερο από τον απαιτούμενο αριθμό (Complement).

Η βέλτιστη λύση αυτού του μοντέλου προσφέρει την καλύτερη δυνατή κάλυψη πτήσεων με το διαθέσιμο πλήρωμα, αν και ορισμένες διαδρομές μπορεί να καλύπτονται περισσότερες φορές από το αναγκαίο.

5.2) Κώδικας Αρχικοποίησης Set-Cover, Set-Partition σε C#, IBM CPLEX

5.2.1) Αρχικοποίηση Set-Partition

Παρατίθεται ο κώδικας αρχικοποίησης του Set-Partition με υπολογιστικό παράδειγμα.

5.2.1.1) Αρχικοποίηση δεικτών, Παραμέτρων

```
#region Parameters, Indexes Initialization
```

```
InputData.T = new int(); // DATES
InputData.I = new int(); // EMPLOYEES
InputData.F = new int(); // ROUTES
InputData.Ri = new Dictionary<int, List<int>>(); // List of Rosters per Employee
InputData.Cij_Hours = new Dictionary<(int, int), double>(); // Cij in hours
InputData.Aijf = new Dictionary<(int, int, int), int>(); // Aijf
#endregion
```

```
InputData.F = InputData.FlightRoutesData.Count;
InputData.I = InputData.Employees.Count;
```

```
int EmployeeCounter = 0;
foreach (var emp in InputData.Employees)
{
    #region Cij,Ri,Aijf Initialization
    //Create a List with the Empty Roster
    List<int> RostersForEmployee_List = new List<int>();
    RostersForEmployee_List.Add(0);

    //Insert into Ri
    InputData.Ri.Add(EmployeeCounter, RostersForEmployee_List);

    //Add the Roster to Cij with Cost = 0
    InputData.Cij_Hours.Add((EmployeeCounter, 0), 0);
    #endregion
    EmployeeCounter++;
}
```

```
int Route_Counter = 0;

foreach (var Route in InputData.FlightRoutesData)
{
    for (int i = 0; i < InputData.I; i++)
    {
        InputData.Aijf[(i, 0, RouteCounter)] = 0;
    }
    int Route_Counter = 0;
}
```

5.2.1.2) Δείκτες, παράμετροι

```
#region Indexes

int I = InputData.I;
int F = InputData.F;
int RiMax = InputData.Ri.Values.Max(list => list.Count);

#endregion

#region Parameters

var h = InputData.RoutesPenalty;
var c = InputData.BoundsPenalty;
Dictionary<int, int> Bf = InputData.RoutesCompl_Dict;
Dictionary<(int, int), double> Cij_Hours = InputData.Cij_Hours;
Dictionary<(int, int, int), int> Aijf = InputData.Aijf;
Dictionary<int, List<int>> Ri = InputData.Ri;

#endregion
//Model Initialization
Cplex model = new Cplex();
```

5.2.1.3) Μεταβλητές απόφασης, αντικειμενική συνάρτηση

```
#region Decision Variables

INumVar[,] X = new INumVar[I, RiMax];
INumVar[] Y = new INumVar[F];

for (int i = 0; i < I; i++)
{
    List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList) ?
                                                                    productList : new
List<int>();

    foreach (int j in RosterPerEmployee_List)
    {
        string varNameX = $"X{i + 1}_{j + 1}";

        X[i, j] = model.NumVar(0, 1, NumVarType.Bool, varNameX);
    }
}
for (int f = 0; f < F; f++)
{
    string varNameY = $"Y{f + 1}";

    Y[f] = model.NumVar(0, 1, NumVarType.Bool, varNameY);
}
#endregion

#region Objective Function

ILinearNumExpr objective = model.LinearNumExpr();

for (int i = 0; i < I; i++)
{
```

5. CREW ROSTERING: ΑΡΧΙΚΟΠΟΙΗΣΗ SET-COVER, SET-PARTITION

```
List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                     ? productList : new List<int>();

foreach (int j in RosterPerEmployee_List)
{
    var Cij_Cost = Cij_Hours[(i, j)] * c;
    objective.AddTerm(Cij_Cost, X[i, j]);
}
}
for (int f = 0; f < F; f++)
{
    objective.AddTerm(h, Y[f]);
}

model.AddMinimize(objective);

#endregion
```

5.2.1.4) Περιορισμοί

```
#region Constraints

// #13. Con13
for (int i = 0; i < I; i++)
{
    ILinearNumExpr expr = model.LinearNumExpr();
    List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                     ? productList : new List<int>();
    foreach (int j in RosterPerEmployee_List)
    {
        expr.AddTerm(1, X[i, j]);
    }
    model.AddEq(expr, 1, "CON2_" + (i + 1));
}

// #14. Con14
for (int f = 0; f < F; f++)
{
    ILinearNumExpr expr = model.LinearNumExpr();
    expr.AddTerm(1, Y[f]);
    for (int i = 0; i < I; i++)
    {
        List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                     ? productList : new List<int>();

        foreach (int j in RosterPerEmployee_List)
        {
            expr.AddTerm(Aijf[(i, j, f)], X[i, j]);
        }
    }
    model.AddEq(expr, Bf[f], "CON3_" + (f + 1));
}

#endregion
```

5. CREW ROSTERING: ΑΡΧΙΚΟΠΟΙΗΣΗ SET-COVER, SET-PARTITION

5.2.1.5) Εξαγωγή Αποτελεσμάτων

```
#region Get Results
// Solve the model
model.Solve();
bool solution = (model.GetStatus() == Cplex.Status.Optimal);
if (solution)
{
    Data.ObjValue = model.ObjValue;

    // Export Model
    model.ExportModel(Path.Combine(relativePath, "CS_Set_Partition.lp"));
    model.ExportModel(Path.Combine(relativePath, "CS_Set_Partition.mps"));
}

model.End();
#endregion
```

5.2.1.6) Υπολογιστικό Παράδειγμα

Στους παρακάτω πίνακες, εικόνες δίνονται τα δεδομένα εισαγωγής του παραδείγματος. Οι πίνακες και οι εικόνες παράχθηκαν μέσω της εφαρμογής.

General Settings

Date From: 04/06/2024 13:01 Date To: 25/06/2024 22:55

Penalty Routes: 100000 Penalty Employee: 20

Position: Captain Problem Type: Set_Partition

H= Penalty Routes =10000

Problem Type = Set Partition

Εικόνα 5.3: Δεδομένα Παραδείγματος Set-Partition (1)

Employee Code	Position	Lower Bound	Upper Bound	Airport Code
CA1	Captain	70	80	Leo_Da
CA2	Captain	70	80	EleftVen
CA3	Captain	70	80	Ber_Brand

I = Αριθμός υπαλλήλων = 3

Εικόνα 5.3: Δεδομένα Παραδείγματος Set-Partition (2)

Route Code	Airport Code	Start Date	End Date	Flight Time	Complement Captain
FR1	Leo_Da	12/06/2024 19:32	14/06/2024 19:32	33.6	1
FR2	EleftVen	12/06/2024 19:32	14/06/2024 19:32	33.6	1
FR3	Ber_Brand	12/06/2024 19:32	14/06/2024 19:32	33.6	1
FR4	Sherem	21/06/2024 04:32	23/06/2024 04:32	33.6	1
FR5	Sherem	05/06/2024 04:35	07/06/2024 04:35	33.6	1
FR6	Leo_Da	23/06/2024 04:35	25/06/2024 04:35	33.6	1

F = Αριθμός FlightRoutes = 6

Bf

Εικόνα 5.4 : Δεδομένα Παραδείγματος Set-Partition (3)

Συνοπτικά οι τιμές των δεδομένων εισαγωγής είναι :

- $I = 3, F = 6$
- $c_{ij} = 0 \forall i, j, a_{ijf} = 0 \forall i, j, f$
- $h = 1000, b_f = 1 \forall f$

$R_i =$

i	R(i)
1	1 Roster χωρίς Routes
2	1 Roster χωρίς Routes
3	1 Roster χωρίς Routes

5.2.1.6.1) Αποτελέσματα: Αρχείο .lp

```

\ENCODING=ISO-8859-1
\Problem name: ILOG.CPLEX

Minimize
obj1: 1000 Y1 + 1000 Y2 + 1000 Y3 + 1000 Y4
      + 1000 Y5 + 1000 Y6

Subject To
CON2_1: X1_1 = 1
CON2_2: X2_1 = 1
CON2_3: X3_1 = 1
CON3_1: Y1 = 1
CON3_2: Y2 = 1
CON3_3: Y3 = 1
CON3_4: Y4 = 1
CON3_5: Y5 = 1
CON3_6: Y6 = 1

Bounds
0 <= X1_1 <= 1
0 <= X2_1 <= 1
0 <= X3_1 <= 1
0 <= Y1 <= 1
0 <= Y2 <= 1
0 <= Y3 <= 1
0 <= Y4 <= 1
0 <= Y5 <= 1
0 <= Y6 <= 1

Binaries
X1_1 X2_1 X3_1 Y1 Y2 Y3 Y4 Y5 Y6

End

```

5.2.1.6.2) Αποτελέσματα: Αρχείο .sol

```

<CPLEXSolution version="1.2">
....
<variables>
<variable name="X1_1" index="0" value="1"/>
<variable name="X2_1" index="1" value="1"/>
<variable name="X3_1" index="2" value="1"/>
<variable name="Y1" index="3" value="1"/>
<variable name="Y2" index="4" value="1"/>
<variable name="Y3" index="5" value="1"/>
<variable name="Y4" index="6" value="1"/>
<variable name="Y5" index="7" value="1"/>
<variable name="Y6" index="8" value="1"/>
</variables>
<objectiveValues>
<objective index="0" name="" value="6000"/>
</objectiveValues>
</CPLEXSolution>

```

5.2.2) Αρχικοποίηση Set-Cover

5.2.2.1) Αρχικοποίηση δεικτών, Παραμέτρων

```
#region Parameters,Indexes Initialization

InputData.T = new int(); // DATES
InputData.I = new int(); // EMPLOYEES
InputData.F = new int(); // ROUTES
InputData.Ri = new Dictionary<int, List<int>>>();
InputData.Cij_Hours = new Dictionary<(int, int), double>();
InputData.Aijf = new Dictionary<(int, int, int), int>();
#endregion

InputData.F = InputData.FlightRoutesData.Count;
InputData.I = InputData.Employees.Count;

int EmployeeCounter = 0;
foreach (var emp in InputData.Employees)
{
    #region Cij,Ri,Aijf Initialization
    //Create a List with the Empty Roster
    List<int> RostersForEmployee_List = new List<int>();
    RostersForEmployee_List.Add(0);

    //Insert into Ri
    InputData.Ri.Add(EmployeeCounter, RostersForEmployee_List);

    //Add the Roster to Cij with Cost = 0
    InputData.Cij_Hours.Add((EmployeeCounter, 0), 0);
    #endregion
    EmployeeCounter++;
}

int Route_Counter = 0;

foreach (var Route in InputData.FlightRoutesData)
{
    for (int i = 0; i < InputData.I; i++)
    {
        InputData.Aijf[(i, 0, RouteCounter)] = 0;
    }
    int Route_Counter = 0;
}

#endregion
```

5.2.2.2) Δείκτες, παράμετροι

```
#region Indexes

int I = InputData.I;
int F = InputData.F;
int RiMax = InputData.Ri.Values.Max(list => list.Count);

#endregionregion

#region Parameters

var h = InputData.RoutesPenalty;
Dictionary<int, int> Bf = InputData.RoutesCompl_Dict;
Dictionary<(int, int, int), int> Aijf = InputData.Aijf;
Dictionary<int, List<int>> Ri = InputData.Ri;

#endregionregion
//Model Initialization
Cplex model = new Cplex();
```

5.2.2.3) Μεταβλητές απόφασης, αντικειμενική συνάρτηση

```
#region Decision Variables
// Decision variables

INumVar[,] X = new INumVar[I, RiMax];
INumVar[] Y = new INumVar[F];

for (int i = 0; i < I; i++)
{
    List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                                ? productList : new
List<int>();

    foreach (int j in RosterPerEmployee_List)
    {
        string varNameX = $"X{i + 1}_{j + 1}";

        X[i, j] = model.NumVar(0, 1, NumVarType.Bool, varNameX);
    }
}
for (int f = 0; f < F; f++)
{
    string varNameY = $"Y{f + 1}";

    Y[f] = model.NumVar(0, 1, NumVarType.Bool, varNameY);
}
#endregionregion

#region Objective Function

ILinearNumExpr objective = model.LinearNumExpr();

for (int f = 0; f < F; f++)
{
    objective.AddTerm(h, Y[f]);
}

model.AddMinimize(objective);
```

5. CREW ROSTERING: ΑΡΧΙΚΟΠΟΙΗΣΗ SET-COVER, SET-PARTITION

#endregion

5.2.2.4) Περιορισμοί

#region Constraints

```
// #13. Con13
for (int i = 0; i < I; i++)
{
    ILinearNumExpr expr = model.LinearNumExpr();
    List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                     ? productList : new List<int>();
    foreach (int j in RosterPerEmployee_List)
    {
        expr.AddTerm(1, X[i,j]);
    }
    model.AddEq(expr, 1, "CON2_" + (i + 1));
}

// #14. Con14
for (int f = 0; f < F; f++)
{
    ILinearNumExpr expr = model.LinearNumExpr();
    expr.AddTerm(1, Y[f]);
    for (int i = 0; i < I; i++)
    {
        List<int> RosterPerEmployee_List = Ri.TryGetValue(i, out var productList)
                                             ? productList : new List<int>();

        foreach (int j in RosterPerEmployee_List)
        {
            expr.AddTerm(Aijf[(i, j, f)], X[i, j]);
        }
    }
    model.AddGe(expr, Bf[f], "CON3_" + (f + 1));
}
```

#endregion

5.2.2.5) Εξαγωγή Αποτελεσμάτων

#region Get Results

```
// Solve the model
model.Solve();
bool solution = (model.GetStatus() == Cplex.Status.Optimal);
if (solution)
{
    Data.ObjValue = model.ObjValue;
    //Export Model
    model.ExportModel(Path.Combine(relativePath, "CS_SetCover.lp"));
    model.ExportModel(Path.Combine(relativePath, "CS_SetCover.mps"));
}
```

#endregion

```
// Dispose of CPLEX objects
model.End();
```

5.2.2.6) Υπολογιστικό Παράδειγμα

Χρησιμοποιήθηκαν τα ίδια δεδομένα που χρησιμοποιήσαμε στο υπολογιστικό παράδειγμα του Set – Partition (5.3.1.6)

Συνοπτικά οι τιμές των δεδομένων εισαγωγής είναι :

- $I = 3, F = 6$
- $c_{ij} = 0 \forall i, j, a_{ijf} = 0 \forall i, j, f$
- $h = 1000, b_f = 1 \forall f$

• $R_i =$

i	R(i)
1	1 Roster χωρίς Routes
2	1 Roster χωρίς Routes
3	1 Roster χωρίς Routes

5.2.2.6.1) Αποτελέσματα: Αρχείο .lp

```
\ENCODING=ISO-8859-1
\Problem name: ILOG.CPLEX

Minimize
  obj1: 1000 Y1 + 1000 Y2 + 1000 Y3 + 1000 Y4 + 1000 Y5
+ 1000 Y6
Subject To
  CON2_1: X1_1 = 1
  CON2_2: X2_1 = 1
  CON2_3: X3_1 = 1
  CON3_1: Y1 >= 1
  CON3_2: Y2 >= 1
  CON3_3: Y3 >= 1
  CON3_4: Y4 >= 1
  CON3_5: Y5 >= 1
  CON3_6: Y6 >= 1
Bounds
  0 <= Y1 <= 1
  0 <= Y2 <= 1
  0 <= Y3 <= 1
  0 <= Y4 <= 1
  0 <= Y5 <= 1
  0 <= Y6 <= 1
  0 <= X1_1 <= 1
  0 <= X2_1 <= 1
  0 <= X3_1 <= 1
Binaries
  Y1 Y2 Y3 Y4 Y5 Y6 X1_1 X2_1 X3_1
End
```

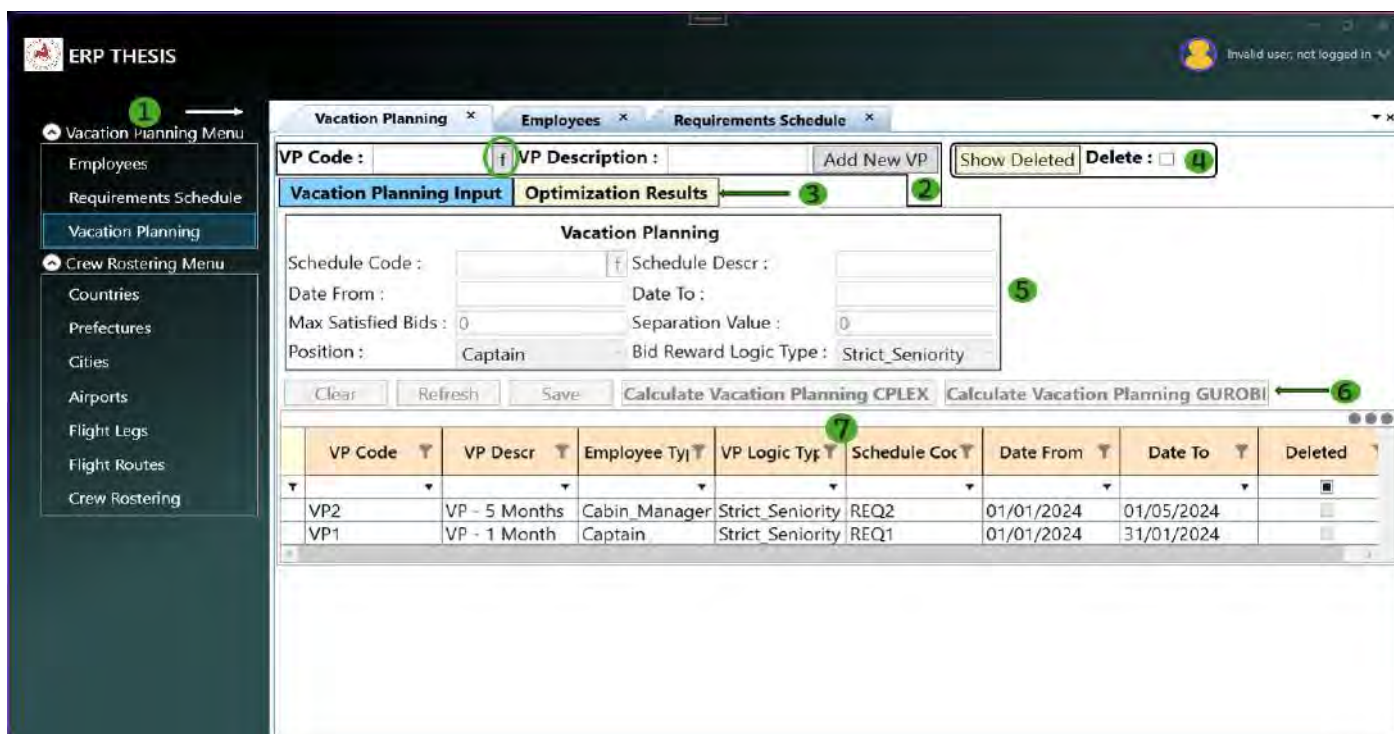
5.2.2.6.2) Αποτελέσματα: Αρχείο .sol

```
CPLEXSolution version="1.2">
....
<variables>
  <variable name="Y1" index="0" value="1"/>
  <variable name="Y2" index="1" value="1"/>
  <variable name="Y3" index="2" value="1"/>
  <variable name="Y4" index="3" value="1"/>
  <variable name="Y5" index="4" value="1"/>
  <variable name="Y6" index="5" value="1"/>
  <variable name="X1_1" index="6" value="1"/>
  <variable name="X2_1" index="7" value="1"/>
  <variable name="X3_1" index="8" value="1"/>
</variables>
<objectiveValues>
  <objective index="0" name="" value="6000"/>
</objectiveValues>
</CPLEXSolution> </objectiveValues>
</CPLEXSolution>
```

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.1) Δομή, Λειτουργίες Εφαρμογής

Αυτή η ενότητα παρέχει μια περιγραφή της εφαρμογής και των βασικών λειτουργιών της. Η εφαρμογή διαθέτει δύο κύρια μενού, ένα για τον προγραμματισμό διακοπών και ένα για την κατανομή πληρώματος, καθιστώντας τη διαχείριση αυτών των διαδικασιών αποτελεσματική και οργανωμένη. Περιλαμβάνει επίσης εργαλεία βελτιστοποίησης για την ανάθεση αδειών και την κατανομή πληρώματος μέσω IBM CPLEX και Gurobi. Οι βασικές λειτουργίες/χαρακτηριστικά που αναλύονται στο κεφάλαιο 6.1 θα γίνουν με βάση την οθόνη “Αρχείο προγραμματισμού διακοπών/Vacation Planning”. Ό,τι αναλυθεί στην κεφάλαιο (6.1) ισχύει για όλες τις οθόνες εκτός από τα Αρχεία Χωρών, Νομών, Πόλεων όπου είναι πιο απλές σε λειτουργίες.



Εικόνα 6.1: Δομή προγράμματος

1) Κατάλογος επιλογής οθονών

Ο χρήστης επιλέγει μια οθόνη από τα 2 διαφορετικά Menu. Η οθόνη εμφανίζεται και προστίθεται στις Καρτέλες (Tab Control) δεξιά από τον αριθμό (1) .

Σύντομη περιγραφή των οθονών για τα 2 διαφορετικά Menu :

Vacation Planning Menu

- Αρχείο υπαλλήλων / Employees: Διαχείριση γενικών πληροφοριών, συνολικές ημέρες για άδεια (Entitlement) και προτιμήσεων για άδεια (Leave Bids) των υπαλλήλων.
- Αρχείο χρονοδιαγράμματος απαιτήσεων/Requirement Schedule: Διαχείριση γενικών πληροφοριών, κατασκευή χρονοδιαγράμματος, με τις ελλείψεις ανά ημέρα (Limit Line) και ανά επάγγελμα (Captain, FO, Cabin Manager, Flight Attendant).
- Αρχείο προγραμματισμού διακοπών/Vacation Planning: Διαχείριση πληροφοριών, δεδομένων εισαγωγής του προβλήματος ανάθεσης αδειών. Επιλογή επίλυσης του προβλήματος μέσω IBM CPLEX ή Gurobi . Εμφάνιση Αποτελεσμάτων.

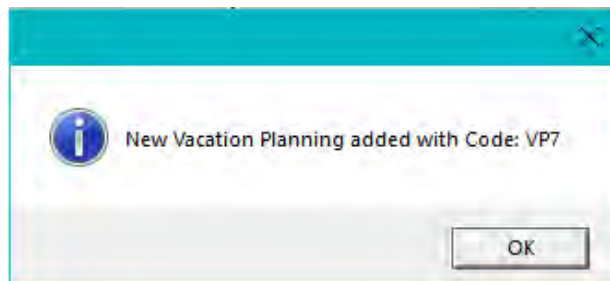
Crew Rostering Menu

- Αρχείο χωρών / Countries: Διαχείριση πληροφοριών σχετικές με τις χώρες.
- Αρχείο νομών / Prefecture: Διαχείριση πληροφοριών σχετικές με τους νομούς.
- Αρχείο πόλεων / City: Διαχείριση πληροφοριών σχετικές με τις πόλεις.
- Αρχείο αεροδρομίων / Airports : Διαχείριση πληροφοριών σχετικές με τα αεροδρόμια.
- Αρχείο πτήσεων/ Flight Legs : Διαχείριση πληροφοριών σχετικές με τις πτήσεις.
- Αρχείο διαδρομών/Flight Routes : Διαχείριση πληροφοριών σχετικές με τις διαδρομές.
- Αρχείο Κατανομής Προσωπικού/Crew Rostering : Διαχείριση πληροφοριών, εισαγωγή δεδομένων και εμφάνιση αποτελεσμάτων για το πρόβλημα των αρχικοποίησης Set-Cover, Set-Partition.

2) Επιλογή, Προσθήκη

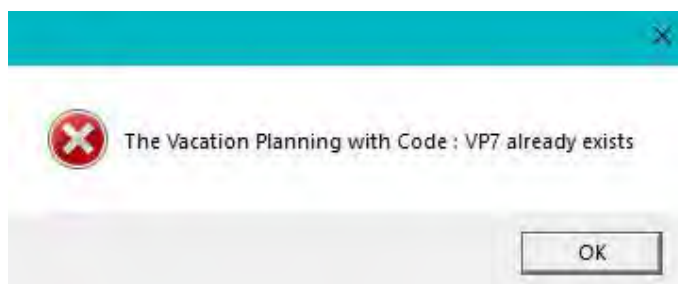
Εισήγαγε μοναδικό κωδικό και περιγραφή και πάτα το κουμπί Add New VP /Προσθήκη .Εναλλακτικά πάτα το κουμπί [f] , που βρίσκεται δίπλα από το κωδικό, ώστε να καλέσεις όλα τα αποθηκευμένα πειράματα που βρίσκονται στη βάση δεδομένων. Τα αποθηκευμένα πειράματα εμφανίζονται στο πίνακα / Data grid που βρίσκεται στο δείκτη (7). Πατώντας διπλό κλικ σε μια γραμμή επιλέγεις ένα πείραμα.

Με την επιτυχή εισαγωγή νέου πειράματος εμφανίζεται το παρακάτω μήνυμα (message box) στην οθόνη .



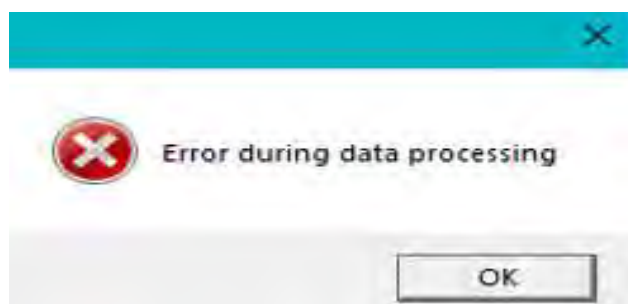
Εικόνα 6.2: Μήνυμα επιτυχούς προσθήκης

Αν ο κωδικός υπάρχει ήδη στη βάση δεδομένων, εμφανίζεται το παρακάτω error μήνυμα.



Εικόνα 6.3: Error μήνυμα προσθήκης

Σε περίπτωση προγραμματιστικού σφάλματος, εμφανίζεται το παρακάτω error μήνυμα.



Εικόνα 6.19: Error, Σφάλμα Εφαρμογής

3) Υπο-Καρτέλες:

Οργάνωση λειτουργιών κάθε οθόνης .Επιλέγοντας καρτέλα το χρώμα της γίνεται μπλέ και εμφανίζεται η αντίστοιχη οθόνη.

4) Διαχείριση Διαγραμμένων :

VP Code	VP Descr	Employee Type	VP Logic Type	Schedule Code	Date From	Date To	Deleted
VP1	VP - 1 Month	Captain	Strict_Seniority	REQ1	01/01/2024	31/01/2024	☐
VP2	VP - 5 Months	Cabin_Manager	Strict_Seniority	REQ2	01/01/2024	01/05/2024	☐
VP3	VP3 CM	Cabin_Manager	Strict_Seniority	Final	01/04/2024	01/05/2024	☒
VP4	VP4 Descr	Captain	Strict_Seniority	REQ-001	01/01/2024	31/01/2024	☒
VP5	VP5 Descr	Captain	Strict_Seniority	REQ1	01/01/2024	31/01/2024	☒

Εικόνα 6.4: Διαχείριση διαγραμμένων

- Delete:

Πατήστε κλικ στο check box και έπειτα πατήστε το κουμπί [save] 'διαγράφει' το πείραμα. Το πείραμα δεν διαγράφεται μόνιμα, απλά ο δείκτης Delete μετατρέπεται σε true και δεν εμφανίζεται στο Data Grid (7).

- Show Deleted:

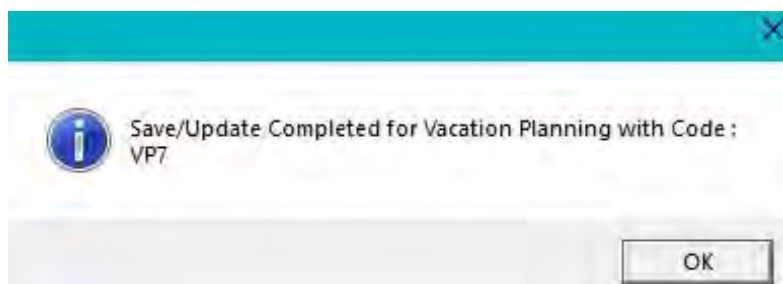
Πατώντας το κουμπί [Show Deleted] και μετά το κουμπί [f] δίπλα από τον κωδικό εμφανίζονται τα «διαγραμμένα» πειράματα στο Data Grid. Μπορείτε να επιλέξετε με διπλό κλικ ένα διαγραμμένο πείραμα, να θέσετε τον δείκτη [delete] ως false, να πατήσετε το κουμπί save και να επαναφέρετε το πείραμα.

5) Πληροφορίες / Δεδομένα:

Εισαγωγή πληροφοριών / Δεδομένων του κάθε πειράματος.

6) Βασικές Εντολές Διαχείρισης δεδομένων:

- Καθαρισμός / Clear: Καθαρίζει όλα τα δεδομένα του πειράματος εκτός από τον κωδικό και την περιγραφή.
- Ανανέωση / Refresh: Επαναφέρει τα δεδομένα του πειράματος, με αυτά που είναι αποθηκευμένα στη βάση.
- Αποθήκευση / Save: Αποθήκευση των δεδομένων στη βάση. Κατά την επιτυχή αποθήκευση εμφανίζεται το παρακάτω μήνυμα (message box) στην οθόνη.



Εικόνα 6.5: Μήνυμα επιτυχής αποθήκευσης, ανανέωσης

4) Calculate Vacation Planning CPLEX / Gurobi : Θα αναλυθεί στο κεφάλαιο (6.4).

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

7) Πίνακας Δεδομένων / Data Grid :

Πατώντας το κουμπί [f] το Data grid εμφανίζει τα δεδομένα στη βάση δεδομένων. Πατώντας διπλό κλικ σε μια γραμμή, επιλέγεται το πείραμα που θέλουμε. Σε περίπτωση πολλαπλών κουμπιών [f], όπως θα δούμε για παράδειγμα στην Οθόνη του Αρχείου Υπαλλήλων το Data grid αλλάζει δυναμικά και εμφανίζει τα δεδομένα που σχετίζονται με το εκάστοτε κουμπί [f].

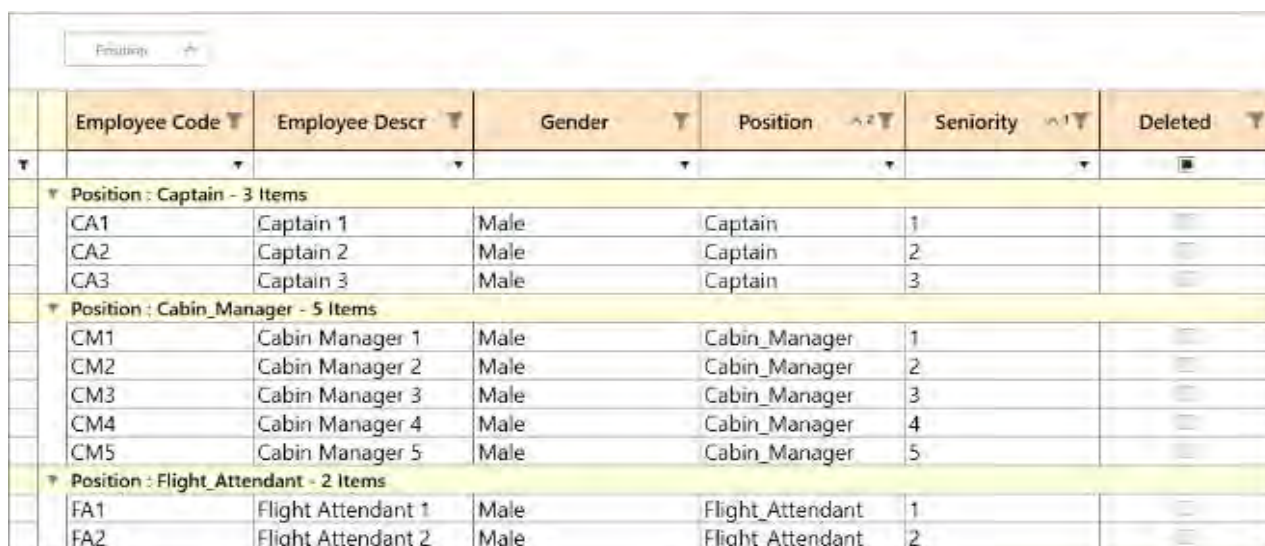
Λειτουργίες Data Grid:

- Ταξινόμηση/Sorting ανά κολόνα/Column:

Πατώντας κλικ σε μια κολόνα το Data grid ταξινομείται με βάση την επιλεγμένη κολόνα. Πατώντας πολλαπλές κολόνες διατηρείται μια σειρά προτεραιότητας στην ταξινόμηση. Όπως βλέπετε στην εικόνα (6.6) ταξινομούμε αρχικά με βάση το Seniority του υπαλλήλου και έπειτα με βάση το Position.

- Ομαδοποίηση / Grouping ανά κολόνα/Column:

Σέρνοντας την μία κολόνα προς τα πάνω τα δεδομένα ομαδοποιούνται. Επιτρέπεται πολλαπλή ομαδοποίηση με σειρά προτεραιότητας. Όπως βλέπετε στην εικόνα (6.6) ομαδοποιούμε με βάση το Position του υπαλλήλου.

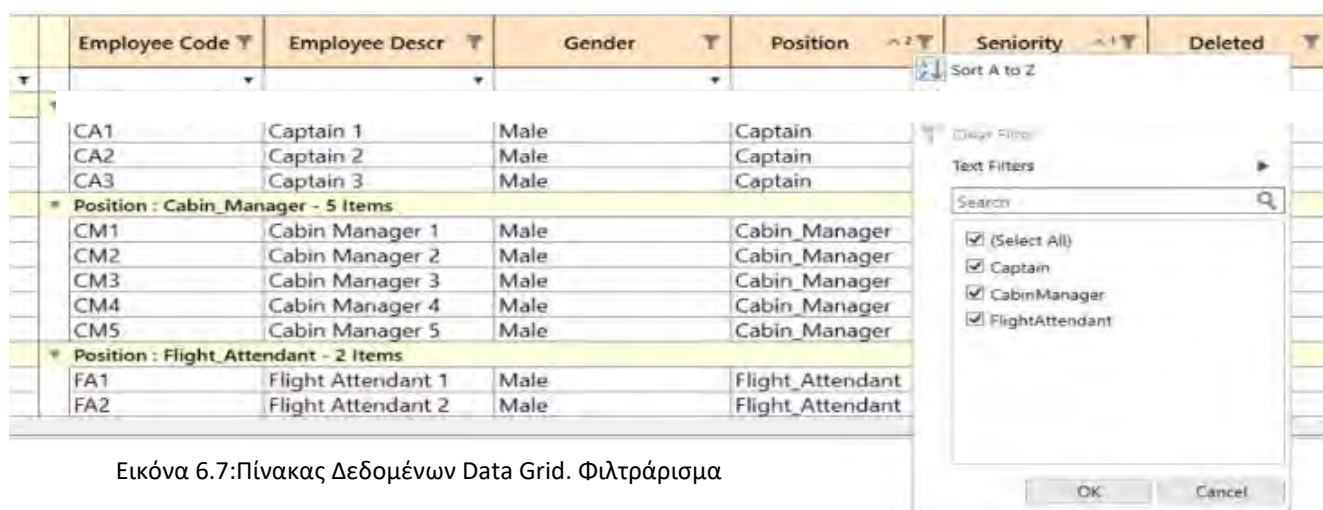


Employee Code	Employee Descr	Gender	Position	Seniority	Deleted
Position : Captain - 3 Items					
CA1	Captain 1	Male	Captain	1	
CA2	Captain 2	Male	Captain	2	
CA3	Captain 3	Male	Captain	3	
Position : Cabin_Manager - 5 Items					
CM1	Cabin Manager 1	Male	Cabin_Manager	1	
CM2	Cabin Manager 2	Male	Cabin_Manager	2	
CM3	Cabin Manager 3	Male	Cabin_Manager	3	
CM4	Cabin Manager 4	Male	Cabin_Manager	4	
CM5	Cabin Manager 5	Male	Cabin_Manager	5	
Position : Flight_Attendant - 2 Items					
FA1	Flight Attendant 1	Male	Flight_Attendant	1	
FA2	Flight Attendant 2	Male	Flight_Attendant	2	

Εικόνα 6.6:Πίνακας Δεδομένων Data Grid. Ταξινόμηση, ομαδοποίηση

- Φιλτράρισμα /Filtering ανά κολόνα/Column:

Μπορούμε να φιλτράρουμε τα δεδομένα μας πατώντας κλικ στο κουμπί  πατώντας τα check και γενικά να τα οργανώσουμε όπως προτιμάμε για την καλύτερη κατανόηση και διαχείριση τους.



Employee Code	Employee Descr	Gender	Position	Seniority	Deleted
CA1	Captain 1	Male	Captain		
CA2	Captain 2	Male	Captain		
CA3	Captain 3	Male	Captain		
Position : Cabin_Manager - 5 Items					
CM1	Cabin Manager 1	Male	Cabin_Manager		
CM2	Cabin Manager 2	Male	Cabin_Manager		
CM3	Cabin Manager 3	Male	Cabin_Manager		
CM4	Cabin Manager 4	Male	Cabin_Manager		
CM5	Cabin Manager 5	Male	Cabin_Manager		
Position : Flight_Attendant - 2 Items					
FA1	Flight Attendant 1	Male	Flight_Attendant		
FA2	Flight Attendant 2	Male	Flight_Attendant		

Clear Filter

Text Filters

Search

☒ (Select All)

☒ Captain

☒ CabinManager

☒ FlightAttendant

OK Cancel

Εικόνα 6.7:Πίνακας Δεδομένων Data Grid. Φιλτράρισμα

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.2) Αρχείο Υπαλλήλων / Employees

Αυτή η ενότητα καλύπτει τη διαχείριση γενικών πληροφοριών και προτιμήσεων αδειών των υπαλλήλων. Περιλαμβάνει την εισαγωγή δεδομένων όπως ονόματα, στοιχεία επικοινωνίας, επαγγελματική θέση κλπ. Στο πλαίσιο του προβλήματος ανάθεσης αδειών, διαχειρίζεται τις άδειες, τις συνολικές ημέρες άδειας και την αρχαιότητα κάθε υπαλλήλου. Στο πλαίσιο κατανομής πληρώματος, διαχειρίζεται τα όρια πτήσης και τις βάσεις / Αεροδρόμια των υπαλλήλων. Η δεύτερη καρτέλα, "Προτιμήσεις Αδειών/Leave Preference", επικεντρώνεται στην εισαγωγή προτιμήσεων αδειών και τη διαχείριση των υπολειπόμενων ημερών άδειας.

6.2.1) 1η Καρτέλα Αρχείου Υπαλλήλων : Γενικές Πληροφορίες

Employees x

Employee Code : CA1 Employee Descr : Captain 1 Add New Employee Show Deleted Delete : ☐

General Information Leave Preference

First Name : Nikos Last Name : Polychronopoulos
Email : npolychr@uth.gr Contact Number : 6945656565
Address : Athinas 5 Date Of Birth : 25/06/2024
Gender : Male Language : 2
Hire Date : 25/06/2024 Total Flight Hours : 100
Position : Captain Seniority : 1
Lower Bound : 70 Upper Bound : 80
Base Airport : Leo_Da
Clear Refresh Save

Employee Code	Employee Des	First Name	Last Name	Gender	Position	Seniority	Deleted
Position : Captain - 3 Items							
CA1	Captain 1	Nikos	Polychronopoulos	Male	Captain	1	
CA2	Captain 2	Xristos	Polychronopoulos	Male	Captain	2	
CA3	Captain 3	Stavroula	Tsiolakaki	Male	Captain	3	
Position : Cabin_Manager - 5 Items							
CM1	Cabin Manager 1	Vasilis	Polychronopoulos	Male	Cabin_Manager	1	
CM2	Cabin Manager 2	Vasilis	Polychronopoulos Sr.	Male	Cabin_Manager	2	
CM3	Cabin Manager 3	Theodoros	Polychronopoulos	Male	Cabin_Manager	3	
CM4	Cabin Manager 4	Nikos	Tsiolakakis	Male	Cabin_Manager	4	
CM5	Cabin Manager 5	Stavros	Tsiolakakis	Male	Cabin_Manager	5	
Position : Flight_Attendant - 2 Items							
FA1	Flight Attendant 1	Rigoula	Polychronopoulou	Male	Flight_Attendant	1	
FA2	Flight Attendant 2	Tasia	Tsiolakaki	Male	Flight_Attendant	2	

Εικόνα 6.8: Αρχείο υπαλλήλων

1) Επιλογή / Προσθήκη : Πληκτρολόγησε Κωδικό, περιγραφή υπαλλήλου και πάτα το κουμπί [Add New Employee]. Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε έναν ήδη υπάρχοντα υπάλληλο από το Data Grid στο (7).

2) Υποκαρτέλες :

- Γενικές Πληροφορίες/General Information:

Διαχείριση πληροφοριών/δεδομένων του υπαλλήλου

- Προτιμήσεις Αδειών/Leave Preference:

Εισαγωγή προτιμήσεων αδειών υπαλλήλου. Διαχείριση υπολειπόμενων ημερών άδειας[Entitlement]

3) Γενικές Πληροφορίες :

- First Name: Όνομα
- Last Name: Επίθετο
- Email
- Contact Number: Τηλέφωνο Επικοινωνίας
- Address: Διεύθυνση
- Gender: Φύλο
- Language: Ξένες Γλώσσες (1 για αγγλικά , 2 για γαλλικά , 3 για γαλλικά και αγγλικά)
- Total Flight Hours: Συνολικές Ώρες πτήσης

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

4) Πληροφορίες σχετικές με το πρόβλημα ανάθεσης αδειών / Vacation Planning:

- Position: Επάγγελμα (Captain, FO, Cabin Manager, Flight Attendant)
- Seniority: Αρχαιότητα / Ιεραρχία υπαλλήλου με το αριθμό 1 να θεωρείται ο αρχαιότερος Μοναδικός αριθμός ανά επάγγελμα.

5) Πληροφορίες σχετικές με το πρόβλημα κατανομής πληρώματος / Crew Rostering:

- Lower Bound: Κατώτερο όριο ωρών πτήσης ανά μήνα
- Upper Bound: Ανώτερο όριο ωρών πτήσης ανά μήνα
- Base Airport: Βασικό αεροδρόμιο εργασίας του υπαλλήλου. Πατώντας του κουμπί [f] εμφανίζονται τα αποθηκευμένα αεροδρόμια στο Data Grid και στη συνέχεια πατώντας διπλό κλικ επιλέγουμε το αεροδρόμιο που θέλουμε.

6) Εντολές διαχείρισης δεδομένων:

- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save

7) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με το κουμπί [f] του Employee Code, είτε από το κουμπί [f] του Base Airport.

6.2.2) 2η Καρτέλα Αρχείου Υπαλλήλων : Προτιμήσεις Αδειών Υπαλλήλου

Leave Bid Code	Priority Level	Bid Type	Date From	Date To	Number Of Days	MIN Number Of Days	MAX Number Of Days	Main Schedule Code
CA1-REQ1-1	1	Specific	20/01/2024	30/01/2024	11	0	0	REQ1
CA1-REQ1-2	2	Non_Specific	01/01/2024	10/01/2024	9	0	0	REQ1
CA1-REQ1-3	3	Min_Max	10/01/2024	22/01/2024	0	9	10	REQ1

Εικόνα 6.9: 2^η Καρτέλα Αρχείου Υπαλλήλων, Leave Preference

1) Εισαγωγή / Διαχείριση προτιμήσεων αδειών :

- Leave Type : Είδος Άδειας
- Bid Priority Level : Προτεραιότητα προτίμησης άδειας
- Μοναδικός ανά άδεια με τον αριθμό 1 να θεωρείται η μέγιστη προτίμηση
- Bid Code: Κωδικός άδειας. Δημιουργείται ως : "Employee Code-Schedule Code-Priority Level".

Διαλέγοντας είδος άδειας (Specific, Non-Specific, Min Max) ,το κουτί (1) αλλάζει .

Leave Type	Bid Priority Level	Bid Code	Date From	Date To
Specific	Specific	CA1-REQ1-4		10/01/2024

Εικόνα 6.10: Εισαγωγή ανά είδος άδειας

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

Παρακάτω βλέπετε τα 3 UI ανάλογα το είδος άδειας.

The image shows three screenshots of the 'Insert/Update Leave Bid' form, each for a different leave type. The top-left form is for 'Specific' leave, with fields for 'Leave Type' (Specific), 'Bid Priority Level' (4), 'Bid Code' (CA1-REQ1-4), 'Date From' (01/01/2024), and 'Date To' (11/01/2024). The top-right form is for 'Non-Specific' leave, with fields for 'Leave Type' (Non-Specific), 'Bid Priority Level' (4), 'Bid Code' (CA1-REQ1-4), 'Date From' (01/01/2024), 'Date To' (10/01/2024), and 'Number Of Days' (8). The bottom form is for 'Min_Max' leave, with fields for 'Leave Type' (Min_Max), 'Bid Priority Level' (4), 'Bid Code' (CA1-REQ1-4), 'Date From' (01/01/2024), 'Date To' (10/01/2024), 'Min' (8), and 'Max' (9). Each form has buttons for 'Refresh', 'Save Bids', 'Add/Modify', and 'Delete Bid'.

Εικόνα 6.11: Επεξεργασία πρότιμησης άδειας

2) Βασικό Πρόγραμμα / Main Schedule :

Εμφανίζεται το βασικό χρονοδιάγραμμα απαιτήσεων. Ο υπάλληλος εισάγει προτιμήσεις αδειών ανάλογα με το εκάστοτε βασικό Χρονοδιάγραμμα. Ο λόγος που έχει ενταχθεί αυτή η λειτουργία είναι ώστε τα δεδομένα εισαγωγής στα πειράματα βελτιστοποίησης ανάθεσης αδειών να επιλέγονται με βάση το χρονοδιάγραμμα και το επάγγελμα.

3) Κατάσταση Αδειών / Leave Status :

- Total: Συνολικές ημέρες άδειας
- Used: Χρησιμοποιημένες ημέρες άδειας
- Current Balance: Υπολειπόμενες ημέρες άδειας ή αλλιώς Entitlement
- Save, Refresh: Κουμπιά Αποθήκευσης, Ανανέωσης

4) Εντολές διαχείρισης προτιμήσεων άδειας :

- Save Bids: Αποθήκευση των προτιμήσεων άδειας στη βάση δεδομένων.
- Delete Bids:

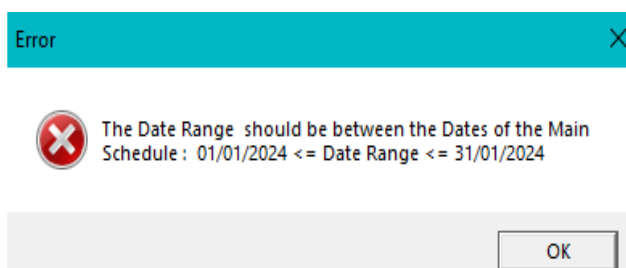
Μόνιμη διαγραφή της επιλεγμένης προτίμησης άδειας από τη βάση. Δεν γίνεται διαχείριση αποθήκευσης των διαγραμμένων όπως έχουμε δει σε προηγούμενα κεφάλαια.

- Add /Modify:

Πρόσθεση άδειας ή ανανέωση υπάρχουσας. Προσοχή, για την ολοκλήρωση την προσθήκης ή ανανέωσης πρέπει να πατήσετε το κουμπί Save ώστε να μεταφερθούν οι αλλαγές στη Βάση.

Σε περίπτωση εισαγωγής λανθασμένων δεδομένων εμφανίζονται τα παρακάτω error μηνύματα στην οθόνη (message box):

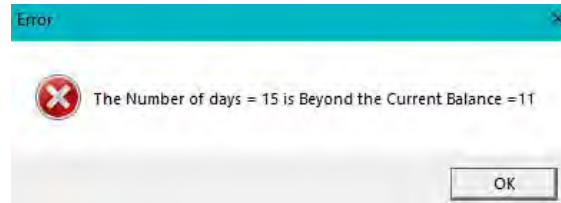
α) Εισαγωγή Date From -Date εκτός του χρονικού ορίζοντα του Βασικού Χρονοδιαγράμματος



Εικόνα 6.12: Εισαγωγή Bid, Error χρονικού ορίζοντα

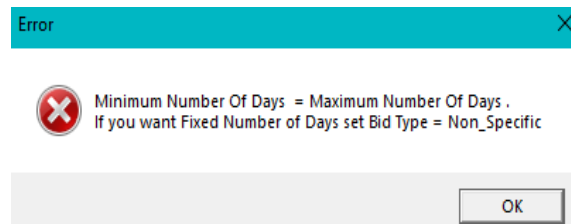
6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

β) Εισαγωγή αριθμού ημερών άδειας μεγαλύτερο από τον υπόλοιπο αριθμό αδειών (Current Balance) του υπαλλήλου.



Εικόνα 6.13: Εισαγωγή Bid, Error αριθμού ημερών

γ) Εισαγωγή είδους άδειας Min Max με min = max αριθμό αδειών. Το error μήνυμα μας προτρέπει σε αυτή την περίπτωση να επιλέξουμε Non-Specific αντί για Min-Max.



Εικόνα 6.14: Εισαγωγή Bid, Error Min = Max

5) Πίνακας Δεδομένων / Data grid :

Εμφανίζονται οι προτιμήσεις άδειας του υπαλλήλου για το βασικό χρονοδιάγραμμα. Πατώντας διπλό κλικ επιλέγεται μια προτίμηση για επεξεργασία.

6.3) Αρχείο Χρονοδιαγράμματος Απαιτήσεων / Requirement Schedule

Αυτή η ενότητα περιλαμβάνει την εισαγωγή και διαχείριση δεδομένων χρονοδιαγράμματος, όπως τον χρονικό ορίζοντα καθώς και τις απαιτήσεις (Limit Line) για κάθε ημέρα και επάγγελμα. Παρέχει εργαλεία για την οργάνωση των χρονοδιαγραμμάτων καθώς και τις βασικές εντολές διαχείρισης δεδομένων.

6.3.1) 1η Καρτέλα Χρονοδιαγράμματος: Γενικές Πληροφορίες

Schedule Code	Schedule Description	Date From	Date To	Limit Line Fixed	Deleted	Main Schedule
REQ1	Req Schedule 1	01/01/2024	31/01/2024	1	☐	☒
REQ2	Req Schedule 2	01/01/2024	01/05/2024	2	☐	☐

Εικόνα 6.15: 1^η Καρτέλα Αρχείου χρονοδιαγράμματος, General Information

1) Επιλογή / Προσθήκη :

Πληκτρολόγησε κωδικό, περιγραφή Χρονοδιαγράμματος και πάτα το κουμπί [Add New schedule].

Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε ένα ήδη υπάρχον χρονοδιάγραμμα από το Data Grid στο (5).

2) Υποκαρτέλες :

- Γενικές Πληροφορίες/General Information:

Διαχείριση πληροφοριών/δεδομένων του χρονοδιαγράμματος.

- Χρονοδιάγραμμα/Schedule:

Κατασκευή χρονοδιαγράμματος. Εισαγωγή Limit Line ανά ημέρα και επάγγελμα.

3) Γενικές Πληροφορίες :

- Date From: Ημερομηνία έναρξης χρονικού ορίζοντα.
- Date To: Ημερομηνία έναρξης χρονικού ορίζοντα.
- Limit Line Fixed: Εισαγωγή default Limit Line.

4) Εντολές διαχείρισης δεδομένων:

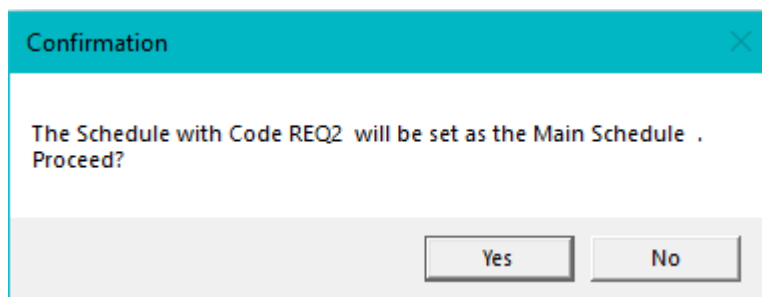
- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save

5) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με το κουμπί [f] του Schedule Code.

6) Βασικό Πρόγραμμα / Main Schedule:

Πατώντας το κουμπί [Main Schedule] θέτουμε το επιλεγμένο χρονοδιάγραμμα ως βασικό. Το Βασικό χρονοδιάγραμμα εμφανίζεται με μπλέ χρώμα στο Data Grid.Εμφανίζεται αντίστοιχο μήνυμα επιβεβαίωσης της εντολής στην οθόνη.



Εικόνα 6.16: Βασικό χρονοδιάγραμμα, Μήνυμα επιβεβαίωσης

6.3.2) 2η Καρτέλα Χρονοδιαγράμματος: Χρονοδιάγραμμα

Position	Date	Limit Line
▲ Position : Captain - 31 Items		
▲ Position : FO - 31 Items		
▲ Position : Cabin_Manager - 31 Items		
▼ Position : Flight_Attendant - 31 Items		
Flight_Attendant	01/01/2024	1
Flight_Attendant	02/01/2024	1
Flight_Attendant	03/01/2024	1
Flight_Attendant	04/01/2024	1
Flight_Attendant	05/01/2024	5
Flight_Attendant	06/01/2024	1

Εικόνα 6.17: 2η Καρτέλα Αρχείου χρονοδιαγράμματος, Schedule

1) Εντολές διαχείρισης δεδομένων:

- Create Hardcoded Data :

Κατασκευάζεται χρονοδιάγραμμα ανά επάγγελμα με χρονικό ορίζοντα από Date From έως Date To που θέσαμε στη Καρτέλα General Information. Το Limit Line κάθε ημέρας ισούται ως το Limit Line Fixed που θέσαμε στη καρτέλα General Information

- Καθαρισμός/Clear : Τίθεται το Limit Line κάθε ημέρας ως το Limit Line Fixed που θέσαμε στην καρτέλα General Information.
- Ανανέωση / Refresh: Επαναφέρει το χρονοδιάγραμμα που βρίσκεται αποθηκευμένο στη βάση.
- Αποθήκευση/ Save : Αποθηκεύεται το πρόγραμμα στη βάση.

2) Πίνακας Δεδομένων / Data Grid:

Εμφανίζεται το χρονοδιάγραμμα με κολόνες το Επάγγελμα, Ημερομηνία, Limit Line.

3) Limit Line:

Μπορούμε να θέσουμε διαφορετικό Limit Line ανά ημέρα και επάγγελμα.

6.4) Αρχείο Προγραμματισμού Διακοπών / Vacation Planning

Αυτή η ενότητα επικεντρώνεται στη διαχείριση του προβλήματος ανάθεσης αδειών των υπαλλήλων και της αρχικοποίησης του Column Generation. Περιλαμβάνει την εισαγωγή δεδομένων για το πρόβλημα ανάθεσης αδειών και την εμφάνιση των αποτελεσμάτων βελτιστοποίησης. Υπάρχουν υποκαρτέλες για την εμφάνιση των χορηγημένων προτιμήσεων άδειας καθώς και άλλων σχετικών δεδομένων. Παρέχει επίσης την δυνατότητα επίλυσης των 2 προβλημάτων μέσω IBM CPLEX ή Gurobi.

6.4.1) 1η Καρτέλα Αρχείου Προγραμματισμού Διακοπών: Γενικές Πληροφορίες

The screenshot shows the 'Vacation Planning' tab with the following elements:

- VP Code:** VP1
- VP Description:** VP - 1 Month
- Schedule Code:** REQ1
- Schedule Descr:** Req Schedule 1
- Date From:** 01/01/2024
- Date To:** 31/01/2024
- Max Satisfied Bids:** 3
- Separation Value:** 3
- Position:** Captain
- Bid Reward Logic Type:** Strict_Seniority
- Buttons:** Add New VP, Show Deleted, Delete, Clear, Refresh, Save, Calculate Vacation Planning CPLEX, Calculate Vacation Planning GUROBI
- Data Grid:**

VP Code	VP Descr	Employee Type	VP Logic Type	Schedule Code	Date From	Date To	Deleted
VP1	VP - 1 Month	Captain	Strict_Seniority	REQ1	01/01/2024	31/01/2024	☐
VP2	VP - 5 Months	Cabin_Manager	Strict_Seniority	REQ2	01/01/2024	01/05/2024	☐

Εικόνα 6.18: 1^η Καρτέλα Αρχείου Προγρ/μου Διακοπών, General Information

1) Επιλογή / Προσθήκη :

Πληκτρολόγησε Κωδικό, περιγραφή Πειράματος ανάθεσης αδειών και πάτα το κουμπί [Add New VP]. Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε ένα ήδη υπάρχον πείραμα από το Data Grid στο (5).

2) Υποκαρτέλες :

- Δεδομένα εισαγωγής στο πρόβλημα ανάθεσης αδειών / Vacation Planning Input:
- Διαχείριση πληροφοριών/δεδομένων του πειράματος
- Αποτελέσματα:

Εμφάνιση Αποτελεσμάτων βελτιστοποίησης. Εμφάνιση των δεδομένων εισαγωγής της αρχικοποίησης του Column Generation και της λύσης της αρχικοποίησης.

3) Γενικές Πληροφορίες :

- Schedule Code: Κωδικός Χρονοδιαγράμματος. Πατώντας το [f] εμφανίζονται τα αποθηκευμένα χρονοδιαγράμματα στο Data Grid. Με διπλό κλικ επιλέγεται χρονοδιάγραμμα και συμπληρώνονται αυτόματα η περιγραφή του Χρονοδιαγράμματος (Schedule Descr) και ο χρονικός ορίζοντας (Date From – Date To).
- Max Satisfied Bids: Μέγιστος αριθμός ικανοποιημένων προτιμήσεων άδειας ανά υπάλληλο .Το αντίστοιχο Bmax στη μορφοποίηση.
- Separation Value: Χρονική διαχωρισμού 2 ικανοποιημένων Bids.Το αντίστοιχο Se στη μορφοποίηση.
- Position: Επάγγελμα Πειράματος (Captain, FO, Cabin Manager, Flight Attendant)
- Bid Reward Logic Type: Λογική Ανάθεσης αδειών (Strict Seniority, Fair Priority, Bid by Bid)

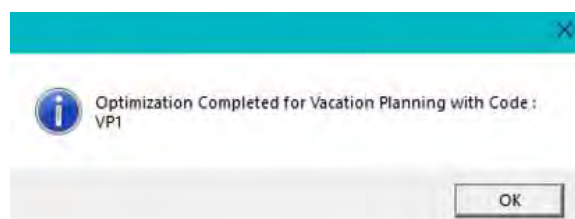
4) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με το κουμπί [f] του VP Code, Schedule Code.

5) Εντολές διαχείρισης δεδομένων:

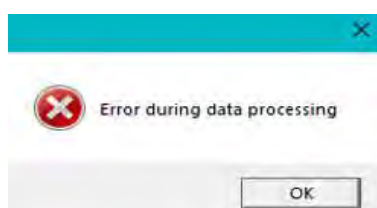
- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save
- Calculate Vacation Planning CPLEX: Επίλυση του προβλήματος μέσω IBM CPLEX
- Calculate Vacation Planning Gurobi: Επίλυση του προβλήματος μέσω Gurobi

Στην επιτυχή βελτιστοποίηση εμφανίζεται το παρακάτω μήνυμα στη οθόνη:



Εικόνα 6.18: Μήνυμα, VP Επιτυχούς βελτιστοποίηση

Σε περίπτωση σφάλματος εμφανίζεται το παρακάτω error μήνυμα στην οθόνη:



Εικόνα 6.19: Error, VP βελτιστοποίηση

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.4.2) 2η Καρτέλα Αρχείου Προγρ/μου Διακοπών: Αποτελέσματα

Vacation Planning x Crew Rostering x Countries x

VP Code : VP1 f VP Description : VP - 1 Month Add New VP Show Deleted Delete : ☐

Vacation Planning Input Optimization Results

Yij Plan Yijrz Plan Xit Plan Entitlement RLLt Plan Column Generation Results 1

Objective Value : 3 VLogic Type : Strict_Seniority Employee Type : Captain 2

	Emp Code ^	Seniority	Bid Code	Bid Type	Priority Level	Date From	Date To	Number Of Days	Min	Max	Granted
▼ Emp Code : CA1 - 3 Items											
	CA1	1	CA1-REQ1-1	Min_Max	1	01/01/2024	10/01/2024	0	8	10	✓
	CA1	1	CA1-REQ1-2	Non_Specific	2	01/01/2024	10/01/2024	9	0	0	■
	CA1	1	CA1-REQ1-3	Specific	3	01/01/2024	10/01/2024	10	0	0	■
▼ Emp Code : CA2 - 3 Items											
	CA2	2	CA2-REQ1-1	Specific	1	20/01/2024	25/01/2024	6	0	0	✓
	CA2	2	CA2-REQ1-2	Min_Max	2	23/01/2024	30/01/2024	0	4	5	■
	CA2	2	CA2-REQ1-3	Specific	3	01/01/2024	10/01/2024	10	0	0	■
▼ Emp Code : CA3 - 2 Items											
	CA3	3	CA3-REQ1-1	Non_Specific	1	22/01/2024	28/01/2024	4	0	0	■
	CA3	3	CA3-REQ1-2	Specific	2	15/01/2024	19/01/2024	5	0	0	✓

Εικόνα 6.20α: 2η Καρτέλα αρχείου προγρ/μου διακοπών , Ανάλυση Οθόνης

1) Υποκαρτέλες :

- Yij Plan:

Εμφάνιση των προτιμήσεων άδειας των υπαλλήλων. Με πράσινο χρώμα εμφανίζονται οι χορηγημένες προτιμήσεις.

- Yijrz Plan:

Εμφάνιση όλων των παραγόμενων επιλογών άδειας από τις προτιμήσεις των υπαλλήλων.

- Xit Plan:

Εμφάνιση του χρονοδιαγράμματος ανά επάγγελμα. Με πράσινο χρώμα εμφανίζονται οι ακριβείς ημέρες που έλαβε άδεια ο κάθε υπάλληλος.

- Entitlement:

Εμφάνιση του συνολικών ημερών άδειας πριν και μετά τη βελτιστοποίηση ανά υπάλληλο.

- RLLt:

Εμφάνιση του υπολειπόμενου Limit Line ανά ημέρα του χρονικού ορίζοντα του επιλεγμένου χρονοδιαγράμματος.

- Column Generation Results:

Εμφανίζεται το αποτέλεσμα της αρχικοποίησης. Η αρχικοποίηση του Column Generation εκτελείται αυτόματα μετά την ανάθεση αδειών.

2) Πληροφορίες Πειράματος:

- Objective Value : Τιμή αντικειμενικής συνάρτησης.

- VLogic Type : Λογική ανάθεσης πειράματος.

- Employee Type : Επάγγελμα πειράματος.

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.4.2.1) 1η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Υij

Emp Code	Seniority	Bid Code	Bid Type	Priority Level	Date From	Date To	Number Of Days	Min	Max	Granted
Emp Code : CA1 - 3 Items										
CA1	1	CA1-REQ1-1	Min_Max	1	01/01/2024	10/01/2024	0	8	10	✓
CA1	1	CA1-REQ1-2	Non_Specific	2	01/01/2024	10/01/2024	9	0	0	✗
CA1	1	CA1-REQ1-3	Specific	3	01/01/2024	10/01/2024	10	0	0	✗
Emp Code : CA2 - 3 Items										
CA2	2	CA2-REQ1-1	Specific	1	20/01/2024	25/01/2024	6	0	0	✓
CA2	2	CA2-REQ1-2	Min_Max	2	23/01/2024	30/01/2024	0	4	5	✗
CA2	2	CA2-REQ1-3	Specific	3	01/01/2024	10/01/2024	10	0	0	✗
Emp Code : CA3 - 2 Items										
CA3	3	CA3-REQ1-1	Non_Specific	1	22/01/2024	28/01/2024	4	0	0	✗
CA3	3	CA3-REQ1-2	Specific	2	15/01/2024	19/01/2024	5	0	0	✓

Εικόνα 6.20β: 2η Καρτέλα αρχείου προγρ/μου διακοπών , Ανάλυση Οθόνης

1) Πίνακας Δεδομένων / Data Grid Yij:

Εμφάνιση των χορηγημένων και μη προτιμήσεων άδειας των υπαλλήλων. Με πράσινο χρώμα εμφανίζονται οι ικανοποιημένες προτιμήσεις.

Ανάλυση κολόνων του Data Grid Yijrz:

- Emp Code: Κωδικός Υπαλλήλου
- Seniority: Ιεραρχία Υπαλλήλου
- Bid Code: Κωδικός προτίμησης άδειας
- Bid Type: Είδος προτίμησης Άδειας (Specific, Non-Specific, Min Max)
- Priority Level: Προτεραιότητα προτίμησης άδειας
- Date From: Ημερομηνία έναρξης προτίμησης άδειας
- Date To: Ημερομηνία λήξης προτίμησης άδειας
- Number of Days: Αριθμός ημερών προτίμησης άδειας. Αν Bid Type = Min Max η τιμή είναι 0
- Min: Ελάχιστος αριθμός ημερών προτίμησης άδειας. Αν Bid Type = Specific ή Non-Specific τότε η τιμή είναι 0
- Max: Μέγιστος αριθμός ημερών προτίμησης άδειας. Αν Bid Type = Specific ή Non-Specific τότε η τιμή είναι 0

2) Χορηγήθηκε / Granted:

Μας δείχνει ποιες προτιμήσεις άδειας χορηγήθηκαν. Αν η τιμή είναι true, η γραμμή γίνεται πράσινη.

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.4.2.2) 2η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Yijrz

Vacation Planning x Crew Rostering x Countries x

VP Code : VP1 f VP Description : VP - 1 Month Add New VP Show Deleted Delete : ☐

Vacation Planning Input Optimization Results

Yij Plan Yijrz Plan Xit Plan Entitlement RLLt Plan Column Generation Results

Objective Value : 3 VLogic Type : Strict_Seniority Employee Type : Captain

Decision Variables Value

Emp Cod ^ 1	Seniority v 2	Bid Code	Bid Type	Priority	Date From	Date To	Number Of Days	Rijr	Yijrz	Granted
▼ Emp Code : CA1 - 9 Items										
CA1	1	CA1-REQ1-1	Min_Max	1	01/01/2024	10/01/2024	10	R111	Y1111	☒
CA1	1	CA1-REQ1-1	Min_Max	1	01/01/2024	09/01/2024	9	R112	Y1121	☐
CA1	1	CA1-REQ1-1	Min_Max	1	02/01/2024	10/01/2024	9	R112	Y1122	☐
CA1	1	CA1-REQ1-1	Min_Max	1	01/01/2024	08/01/2024	8	R113	Y1131	☐
CA1	1	CA1-REQ1-1	Min_Max	1	02/01/2024	09/01/2024	8	R113	Y1132	☐
CA1	1	CA1-REQ1-1	Min_Max	1	03/01/2024	10/01/2024	8	R113	Y1133	☐
CA1	1	CA1-REQ1-2	Non_Specific	2	01/01/2024	09/01/2024	9	R121	Y1211	☐
CA1	1	CA1-REQ1-2	Non_Specific	2	02/01/2024	10/01/2024	9	R121	Y1212	☐
CA1	1	CA1-REQ1-3	Specific	3	01/01/2024	10/01/2024	10	R131	Y1311	☐
▲ Emp Code : CA2 - 11 Items										
> ▲ Emp Code : CA3 - 5 Items										

Εικόνα 6.21: 2η Καρτέλα αρχείου προγρ/μου διακοπών , Yijrz Plan

1) Πίνακας Δεδομένων / Data Grid Yijrz:

Εμφάνιση όλων των παραγόμενων επιλογών άδειας από τις προτιμήσεις των υπαλλήλων. Με πράσινο χρώμα εμφανίζονται οι ακριβείς άδειες που χορηγήθηκαν.

Ανάλυση κολόνων του Data Grid Yijrz:

- Emp Code: Κωδικός Υπαλλήλου
- Seniority: Ιεραρχία Υπαλλήλου
- Bid Code: Κωδικός προτίμησης άδειας
- Bid Type: Είδος προτίμησης Άδειας (Specific, Non-Specific, Min Max)
- Priority Level: Προτεραιότητα προτίμησης άδειας
- Date From: Ημερομηνία έναρξης παραγόμενης άδειας
- Date To: Ημερομηνία λήξης παραγόμενης άδειας
- Number of Days: Αριθμός ημερών παραγόμενης άδειας
- Rijr: Όνομα Μεταβλητής Απόφασης Rijr
- Yijrz: Όνομα Μεταβλητής Απόφασης Yijrz
- Granted: Απεικόνιση της δυαδικής τιμής των Rijr, Yijrz. Όταν η Granted είναι True η γραμμή γίνεται πράσινη.

6.4.2.3) 3η Υποκαρτέλα Αποτελεσμάτων: Πλάνο Xit

Employee Code ^1	Seniority	Date ^2	Xit	Granted
Employee Code : CA1 - 31 Items				
CA1	1	01/01/2024	X11	✓
CA1	1	02/01/2024	X12	✓
CA1	1	03/01/2024	X13	✓
CA1	1	04/01/2024	X14	✓
CA1	1	05/01/2024	X15	✓
CA1	1	06/01/2024	X16	✓
CA1	1	07/01/2024	X17	✓
CA1	1	08/01/2024	X18	✓
CA1	1	09/01/2024	X19	✓
CA1	1	10/01/2024	X110	✓
CA1	1	11/01/2024	X111	
CA1	1	12/01/2024	X112	
CA1	1	13/01/2024	X113	
CA1	1	14/01/2024	X114	
CA1	1	15/01/2024	X115	

Εικόνα 6.22: 2η Καρτέλα αρχείου προγράμμου διακοπών , Xit Plan

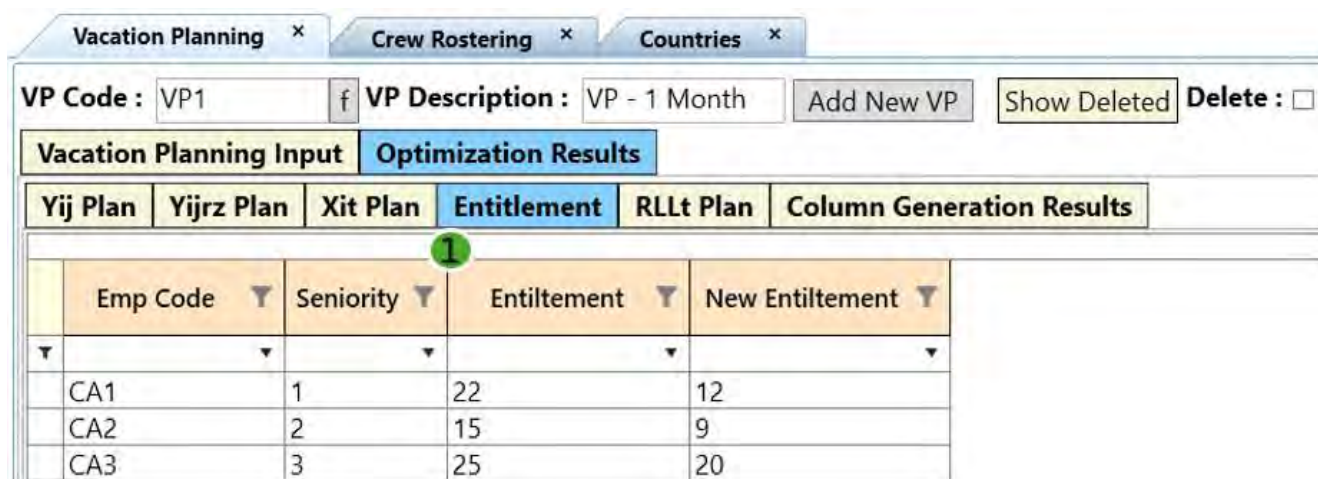
1) Πίνακας Δεδομένων / Data Grid Xit:

Εμφάνιση του χρονοδιαγράμματος ανά επάγγελμα. Με πράσινο χρώμα εμφανίζονται οι ακριβείς ημέρες που έλαβε για άδεια ο κάθε υπάλληλος.

Ανάλυση κολόνων του Data Grid Xit:

- Employee Code: Κωδικός Υπαλλήλου
- Seniority: Ιεραρχία Υπαλλήλου
- Date : Ημέρα
- Xit : Όνομα Μεταβλητής Απόφασης Xit
- Granted: Απεικόνιση της δυαδικής τιμής της Xit. Όταν η Granted είναι True η γραμμή γίνεται πράσινη.

6.4.2.4) 4η Υποκαρτέλα Αποτελεσμάτων: Entitlement



VP Code : VP1 VP Description : VP - 1 Month Add New VP Show Deleted Delete : ☐

Vacation Planning Input Optimization Results

Yij Plan Yijrz Plan Xit Plan Entitlement RLLt Plan Column Generation Results

Emp Code	Seniority	Entitlement	New Entitlement
CA1	1	22	12
CA2	2	15	9
CA3	3	25	20

Εικόνα 6.23: 2^η Καρτέλα αρχείου προγράμμου διακοπών, Entitlement

1) Πίνακας Δεδομένων / Data Grid Entitlement:

Εμφάνιση του συνολικών ημερών άδειας πριν και μετά τη βελτιστοποίηση ανά υπάλληλο. Οι υπολειπόμενες ημέρες άδειας (New Entitlement) εισάγονται ως οι τιμές της παραμέτρου REi στην αρχικοποίηση του Column Generation.

Ανάλυση κολόνων του Data Grid Entitlement:

- Employee Code: Κωδικός Υπαλλήλου
- Seniority: Ιεραρχία Υπαλλήλου
- Entitlement : Συνολικές ημέρες άδειας ανά υπάλληλο πριν την βελτιστοποίηση ανάθεσης αδειών
- New Entitlement : Υπολειπόμενες ημέρες άδειας ανά υπάλληλο μετά την βελτιστοποίηση ανάθεσης αδειών.

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.4.2.5) 4η Υποκαρτέλα Αποτελεσμάτων: RLLt Plan

Vacation Planning x

VP Code : VP1 f VP Description : VP - 1 Month Add New VP Show Delete

Vacation Planning Input Optimization Results

Yij Plan Yijrz Plan Xit Plan Entitlement RLLt Plan Column Generation Results

RLL	Date	LimitLine
RLL1	01/01/2024	0
RLL2	02/01/2024	0
RLL3	03/01/2024	0
RLL4	04/01/2024	0
RLL5	05/01/2024	0
RLL6	06/01/2024	0
RLL7	07/01/2024	0
RLL8	08/01/2024	0
RLL9	09/01/2024	0
RLL10	10/01/2024	0
RLL11	11/01/2024	1
RLL12	12/01/2024	1

Εικόνα 6.24: 2η Καρτέλα αρχείου προγρ/μου διακοπών , RLLt Plan

1) Δεδομένων / Data Grid RLLt:

Εμφάνιση του υπολειπόμενου Limit Line ανά ημέρα του χρονικού ορίζοντα του επιλεγμένου χρονοδιαγράμματος. Τα παραγόμενα υπολειπόμενα Limit Lines ανά ημέρα εισάγονται ως την παράμετρο RLL στην αρχικοποίηση του Column Generation.

Ανάλυση κολόνων του Data Grid RLLt:

- RLL: Όνομα γραμμής
- Date: Ημερομηνία
- Limit Line : Υπολειπόμενο Limit Line.

6.4.2.6) 5η Υποκαρτέλα Αποτελεσμάτων: Column Generation Results

Vacation Planning x Crew Rostering x Countries x

VP Code : VP1 f VP Description : VP - 1 Month Add New VP Show Deleted Delete : ☐

Vacation Planning Input Optimization Results

Yij Plan Yijrz Plan Xit Plan Entitlement RLLt Plan Column Generation Results

Objective Value : 41000 1

Εικόνα 6.25: 2η Καρτέλα αρχείου προγρ/μου διακοπών , Column Generation Results

1) Τιμή Αντικειμενικής Συνάρτησης / Objective Value :

Εμφάνιση της τιμής της Αντικειμενικής συνάρτησης της Αρχικοποίησης. Ισούται με τον άθροισμα των υπολειπόμενων ημερών άδειας των υπαλλήλων x 1000 (Penalty).

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

6.5) Αρχεία Χωρών, Νομών, Πόλεων / Country, Prefecture, City

Αυτή η ενότητα καλύπτει τη διαχείριση δεδομένων χωρών, νομών και πόλεων. Περιλαμβάνει την εισαγωγή και αποθήκευση πληροφοριών για τις χώρες, τους νομούς και τις πόλεις καθώς και τη διαχείριση των διαγραμμένων εγγραφών.

6.5.1) Αρχεία Χωρών, Νομών, Πόλεων: Περιγραφή

Αρχείο Χωρών



Country Code	Country Descr	Deleted
Gr	Greece	☐
Ger	Germany	☐
Fr	France	☐
Ita	Italy	☐
Swe	Sweden	☐
Hun	Hungary	☐
Den	Denmark	☐
Por	Portugal	☐
Ru	Russia	☐
Ukr	Ukraine	☐
Uk	United Kingdom	☐

Εικόνα 6.26: Αρχείο χωρών, Countries

Αρχείο Χωρών

1) Εντολές διαχείρισης δεδομένων:

- Ανανέωση / Refresh
- Αποθήκευση/ Save
- Εμφάνιση 'Διαγραμμένων' / Show Deleted

2) Πίνακας δεδομένων / Data Grid:

- Country Code: Κωδικός Χώρας
- Country Description: Περιγραφή Χώρας
- Deleted: Διαγραμμένο

Prefecture Code	Prefecture Descr	Country Code	Deleted
Brand	Brandenburg	Ger	☐
Hes	Hesse	Ger	☐
Lazio	Lazio Region	Ita	☐
MosOb	Moscow Oblast	Ru	☐
ParReg	Paris Region	Fr	☐
Upp	Uppland	Swe	☐

CountryId	CountryCode	Country
1	Gr	Greece
2	Ger	Germany
3	Fr	France
6	Ita	Italy
17	Swe	Sweden
18	Hun	Hungary
19	Den	Denmark
21	Por	Portugal
22	Ru	Russia

Εικόνα 6.27: Αρχείο νομών, Prefectures

City Code	City Descr	Prefecture Code	Deleted
Ber	Berlin	Brand	☐
Par	Paris	ParReg	☐
Rom	Rome	Lazio	☐
Stock	Stockholm	Upp	☐
Mos	Moscow	MosOb	☐

PrefId	de	PrefD
8	ARK	Arkadia
9	MES	Messinia
10	LAK	Laconia
11	ARG	Argolis
12	ILIA	Ilia
13	KZN	Kozanis
14	KVL	Kavalas
15	XNT	Xanthias
16	KOR	Korinthias

Εικόνα 6.28: Αρχείο πόλεων, Cities

Αρχείο Νομών**2) Πίνακας δεδομένων / Data Grid:**

- Prefecture Code: Κωδικός νομού
- Prefecture Description: Περιγραφή νομού
- Country Code: Κωδικός χώρας
- Deleted: Διαγραμμένο

3) Κωδικός Χώρας:

Εμφανίζονται όλες οι αποθηκευμένες χώρες στο αναδυόμενο Data grid . Με διπλό κλικ επιλέγεις την χώρα του νομού ανά γραμμή.

Αρχείο Πόλεων**2) Πίνακας δεδομένων / Data Grid:**

- City Code: Κωδικός πόλης
- City Description: Περιγραφή πόλης
- Prefecture Code: Κωδικός νομού
- Deleted: Διαγραμμένο

3) Κωδικός Νομού:

Εμφανίζονται όλοι οι αποθηκευμένοι νομοί στο αναδυόμενο Data grid . Με διπλό κλικ επιλέγεις τον νομό της πόλης ανά γραμμή.

6.6) Αρχείο Αεροδρομίων / Airports

Αυτή η ενότητα αφορά τη διαχείριση πληροφοριών αεροδρομίων. Περιλαμβάνει την εισαγωγή και διαχείριση δεδομένων αεροδρομίων καθώς και βασικές εντολές.

6.6.1) 1η Καρτέλα Αρχείου Αεροδρομίων: Γενικές Πληροφορίες

The screenshot shows the 'Airports' application window. At the top, there's a tab labeled 'Airports'. Below it, the 'General Information' tab is selected. The form contains the following elements:

- 1** Airport Code: EleftVen (with a focus icon 'f')
- 1** Airport Descr: Elefth_Venizelos
- 2** General Information tab (highlighted with a blue bar)
- 3** City Code: Ath (with a focus icon 'f')
- 4** Buttons: Clear, Refresh, Save
- 5** Data Grid (highlighted with a green circle)

Airport Code	Airport Descr	City Code	City Description	Prefecture	Country	Deleted
Leo_Da	Leonardo_Davinci	Rom	Rome	Lazio Region	Italy	☐
> EleftVen	Elefth_Venizelos	Ath	Athens	Attica	Greece	☐
Ber_Brand	Berlin_BrandenBurng	Ber	Berlin	Brandenburg	Germany	☐
Sherem	Sheremetyevo	Mos	Moscow	Moscow Oblast	Russia	☐

Εικόνα 6.29: 1η Καρτέλα αρχείου αεροδρομίων , General Information

1) Επιλογή / Προσθήκη :

Πληκτρολόγησε Κωδικό, περιγραφή αεροδρομίου και πάτα το κουμπί [Add New Airport]. Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε ένα ήδη υπάρχον αεροδρόμιο από το Data Grid στο (3).

2) Υποκαρτέλες :

- Γενικές Πληροφορίες / General Information: Διαχείριση πληροφοριών/δεδομένων του αεροδρομίου .

3) Γενικές Πληροφορίες:

- City Code: Κωδικός Πόλης.

4) Εντολές διαχείρισης δεδομένων:

- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save

5) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με τα κουμπιά [f] του Airport Code, City Code.

6.7) Αρχείο πτήσεων / Flight Legs

Αυτή η ενότητα καλύπτει τη διαχείριση πληροφοριών πτήσεων. Περιλαμβάνει την εισαγωγή και διαχείριση δεδομένων για τις πτήσεις, όπως κωδικούς αεροδρομίων αναχώρησης και προορισμού, ημερομηνίες και ώρες.

6.7.1) 1η Καρτέλα Αρχείου πτήσεων: Γενικές Πληροφορίες

Εικόνα 6.30: 1η Καρτέλα αρχείου πτήσεων , General Information

1) Επιλογή / Προσθήκη:

Πληκτρολόγησε Κωδικό, περιγραφή της διαδρομής και πάτα το κουμπί [Add New Route]. Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε μια ήδη υπάρχουσα πτήση από το Data Grid στο (6).

2) Υποκαρτέλες:

- Γενικές Πληροφορίες/General Information:

3) Γενικές Πληροφορίες:

- Airport Code From: Κωδικό αεροδρομίου αναχώρησης.
- Airport Code To: Κωδικός αεροδρομίου προορισμού.
- Start Date: Ημερομηνία αναχώρησης της πτήσης.
- End Date: Ημερομηνία άφιξης της πτήσης.

4) Εργαλείο διαχείρισης ημερομηνίας και ώρας:

Σύστημα καταχώρισης ημερομηνίας και ώρας. Η ώρα εισάγεται σε στρατιωτική μορφή (00:00 έως 24:00).

5) Εντολές διαχείρισης δεδομένων:

- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save

6) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με το κουμπί [f] του Flight Leg Code, Airport Code From, Airport Code To.

6.8) Αρχείο διαδρομών / Flight Routes

Αυτή η ενότητα επικεντρώνεται στη διαχείριση πληροφοριών διαδρομών. Περιλαμβάνει την εισαγωγή και διαχείριση δεδομένων διαδρομών, όπως κωδικούς αεροδρομίων, συνολικός χρόνος διαδρομής, χρόνος πτήσης και χρόνος εδάφους. Παρέχει επίσης τη διαχείριση των απαιτούμενων θέσεων ανά επάγγελμα.

6.8.1) 1η Καρτέλα Αρχείου διαδρομών: Γενικές Πληροφορίες

Flight Routes

Flight Route Code : FR1 f Flight Route Descr : FR1 Descr Add New Route Show Deleted Delete : ☐

General Information Flight Legs

Start Date : 12/06/2024 19:32 End Date : 14/06/2024 19:32

Complement Captain : 1 Complement FO : 1

Complement Cabin Manager : 1 Complement Flight Attendant : 3

Airport Code : Leo_Da f Total Time : 48

Flight Time : 33.6 Ground Time : 14.4

Clear Refresh Save

Flight Code ^	Flight Descr	Airport	Start Date	End Date	Deleted
> FR1	FR1 Descr	Leo_Da	12/06/2024 19:32	14/06/2024 19:32	☐
FR2	FR2 Descr	EleftVen	12/06/2024 19:32	14/06/2024 19:32	☐
FR3	FR3 Descr	Ber_Brand	12/06/2024 19:32	14/06/2024 19:32	☐
FR4	FR4 Descr	Sherem	21/06/2024 04:32	23/06/2024 04:32	☐

Εικόνα 6.31: 1η Καρτέλα αρχείου διαδρομών , General Information

1) Επιλογή / Προσθήκη :

Πληκτρολόγησε Κωδικό, περιγραφή διαδρομής και πάτα το κουμπι [Add New Schedule]. Εναλλακτικά πάτα το κουμπι [f] δίπλα από τον κωδικό και επέλεξε μία ήδη υπάρχουσα διαδρομή από το Data Grid στο (5).

2) Υποκαρτέλες :

- Γενικές Πληροφορίες/General Information:
Διαχείριση πληροφοριών/δεδομένων του διαδρομής.
- Πτήσεις / Flight Legs:
Δεν έχει κατασκευαστεί. Μελλοντικά θα εμφανίζει τις πτήσεις της διαδρομής.

3) Γενικές Πληροφορίες :

- Airport Code: Κωδικός αεροδρομίου της διαδρομής.
- Start Date: Ημερομηνία έναρξης της διαδρομής.
- End Date: Ημερομηνία λήξης της διαδρομής.
- Total Time: Συνολικός χρόνος διαδρομής, περιλαμβάνονται όλες οι φάσεις της πτήσης.
- Flight Time: Χρόνος που το αεροσκάφος βρίσκεται στον αέρα.
- Ground Time: Χρόνος που το αεροσκάφος βρίσκεται στο έδαφος.
- Complement Captain: Ο αριθμός των κυβερνητών που απαιτούνται για την διαδρομή.
- Complement FO: Ο αριθμός των συγκυβερνητών που απαιτούνται για την διαδρομή.
- Complement Cabin Manager: Ο αριθμός των υπευθύνων καμπίνας που απαιτούνται για την διαδρομή.
- Complement Flight Attendant: Ο αριθμός των αεροσυνοδών που απαιτούνται για την διαδρομή.

4) Εντολές διαχείρισης δεδομένων:

- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save

5) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με τα κουμπι [f] του Flight Route Code, Airport Code.

6.9) Αρχείο κατανομής πληρώματος / Crew Rostering

Αυτή η ενότητα καλύπτει τη διαχείριση δεδομένων για την κατανομή πληρώματος. Περιλαμβάνει την εισαγωγή κωδικού και περιγραφής πειράματος κατανομής, την εμφάνιση υπαλλήλων και διαδρομών πειράματος, και την εκτέλεση της αρχικοποίησης. Περιλαμβάνει επίσης τη δυνατότητα επιλογής μεταξύ Set Cover και Set Partition, καθώς και τη διαχείριση ποινών για τις διαδρομές και τους υπαλλήλους.

6.9.1) 1η Καρτέλα Αρχείου κατανομής πληρώματος: Γενικές Πληροφορίες

CS ID	CS Code	CS Descr	Date From	Date To	Position	Is Deleted
1	CS1	CS1 Descr	01/06/2024 13:01	30/06/2024 22:55	Captain	☐
2	CS2	CS2 Descr	03/06/2024 01:25	15/06/2024 11:55	Cabin_Manager	☐
> 3	CS3	CS3 Descr	01/06/2024 13:01	31/05/2024 22:55	Captain	☐

Εικόνα 6.32: 1η Καρτέλα αρχείου κατανομής πληρώματος , General Information

1) Επιλογή / Προσθήκη :

Πληκτρολόγησε Κωδικό, περιγραφή Πειράματος κατανομής πληρώματος και πάτα το κουμπί [Add New Crew Rostering]. Εναλλακτικά πάτα το κουμπί [f] δίπλα από τον κωδικό και επέλεξε ένα ήδη υπάρχον πείραμα από το Data Grid στο (5).

2) Υποκαρτέλες :

- Γενικές Πληροφορίες/General Information:

Διαχείριση πληροφοριών/δεδομένων εισαγωγής του πειράματος. Εμφάνιση υπαλλήλων, διαδρομών πειράματος. Εκτέλεση της αρχικοποίησης

- Optimization Results:

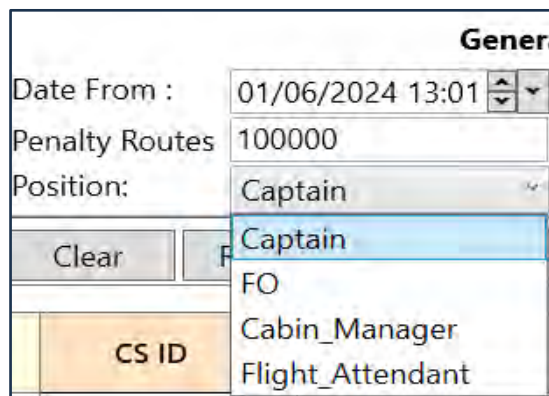
Κατασκευή χρονοδιαγράμματος. Εισαγωγή Limit Line ανά ημέρα και επάγγελμα.

3) Γενικές Πληροφορίες :

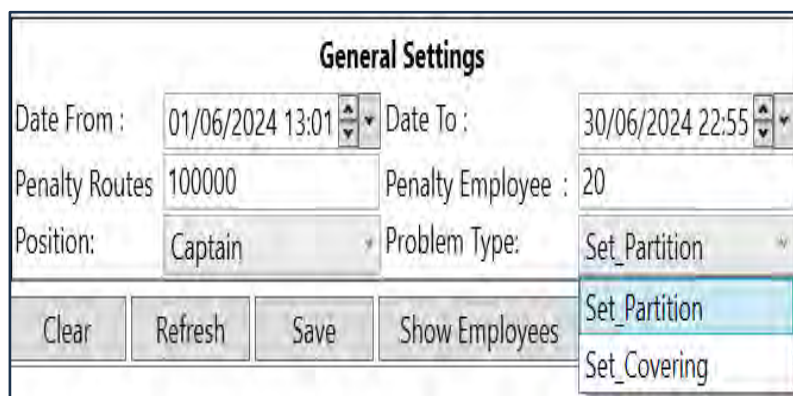
- Date From: Ημερομηνία έναρξης χρονικού ορίζοντα.
- Date To: Ημερομηνία λήξης του χρονικού ορίζοντα.
- Penalty Routes: Η τιμή του penalty για κάθε διαδρομή που παραμένει ανεκπλήρωτη. Στη μορφοποίηση είναι η παράμετρος 'h'
- Penalty Employee: Η τιμή του penalty για κάθε ώρα απόκλισης από τα όρια Lower Bound, Upper Bound κάθε υπαλλήλου. Στη μορφοποίηση είναι η παράμετρος 'c'
- Position: Επάγγελμα υπαλλήλου (Captain, FO, Cabin Manager, Flight Attendant)

6. ΕΓΧΕΙΡΙΔΙΟ ΕΦΑΡΜΟΓΗΣ

- Problem Type: Set Cover ή Set Partition.



Εικόνα 6.33: Κατανομή πληρώματος, Position

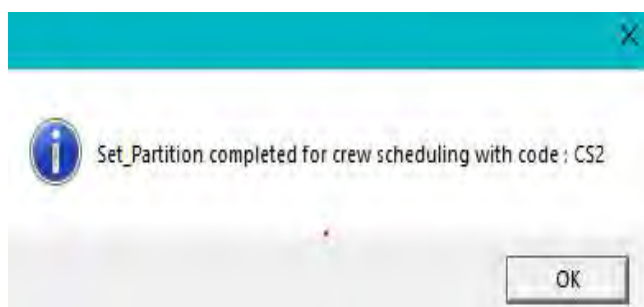


Εικόνα 6.34: Κατανομή πληρώματος, Problem Type

4) Εντολές διαχείρισης δεδομένων:

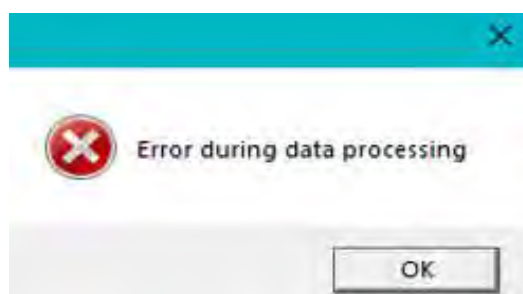
- Καθαρισμός/Clear
- Ανανέωση / Refresh
- Αποθήκευση/ Save
- Εμφάνιση υπαλλήλων / Show Employees: Εμφάνιση υπαλλήλων με βάση το επιλεγμένο επάγγελμα στο Data Grid (5).
- Εμφάνιση διαδρομών / Show Flight Routes: Εμφάνιση διαδρομών εντός του χρονικού ορίζοντα στο Data Grid (5).
- Calculate Crew Rostering: Επίλυση της αρχικοποίησης της κατανομής πληρώματος μέσω IBM CPLEX.

Μήνυμα Βελτιστοποίησης: Επιτυχία



Εικόνα 6.35: Κατανομή πληρώματος,
Μήνυμα Επιτυχούς Βελτιστοποίησης

Μήνυμα Βελτιστοποίησης: Αποτυχία

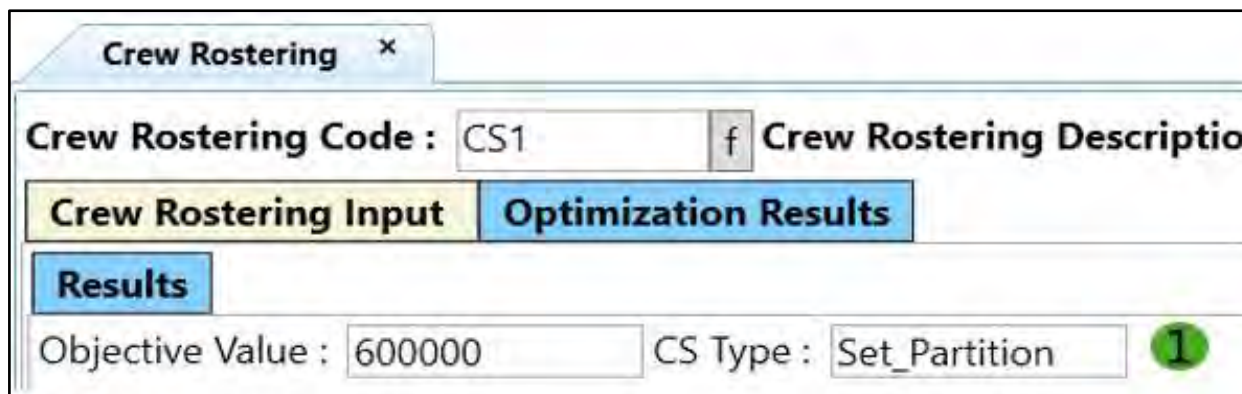


Εικόνα 6.36: Κατανομή πληρώματος,
Error μήνυμα βελτιστοποίησης

5) Πίνακας Δεδομένων / Data Grid:

Ενεργοποιείται με το κουμπί [f] του Crew Rostering Code ή με τα κουμπιά [Show Employees], [Show Flight Routes].

6.9.2) 2η Καρτέλα Αρχείου κατανομής πληρώματος: Αποτελέσματα



The screenshot displays the 'Crew Rostering' application window. At the top, there is a tab labeled 'Crew Rostering' with a close button. Below the tab, the 'Crew Rostering Code' is set to 'CS1' and the 'Crew Rostering Description' is partially visible. The main area contains two tabs: 'Crew Rostering Input' and 'Optimization Results'. The 'Optimization Results' tab is active, showing a 'Results' sub-tab. Below this, the 'Objective Value' is displayed as '600000' and the 'CS Type' is 'Set_Partition'. A green circle with the number '1' is located to the right of the 'CS Type' field.

Εικόνα 6.37: Αποτελέσματα Αρχικοποίησης Κατανομής Πληρώματος

1) Τιμή Αντικειμενικής Συνάρτησης / Objective Value :

- Εμφάνιση της τιμής της αντικειμενικής συνάρτησης της Αρχικοποίησης. Ισούται με τον αριθμό των διαδρομών x Penalty Routes.
- Εμφάνιση του τύπου της επιλυμένης μορφοποίησης.

7.1) Αρχιτεκτονική, MVVM Θεωρία.

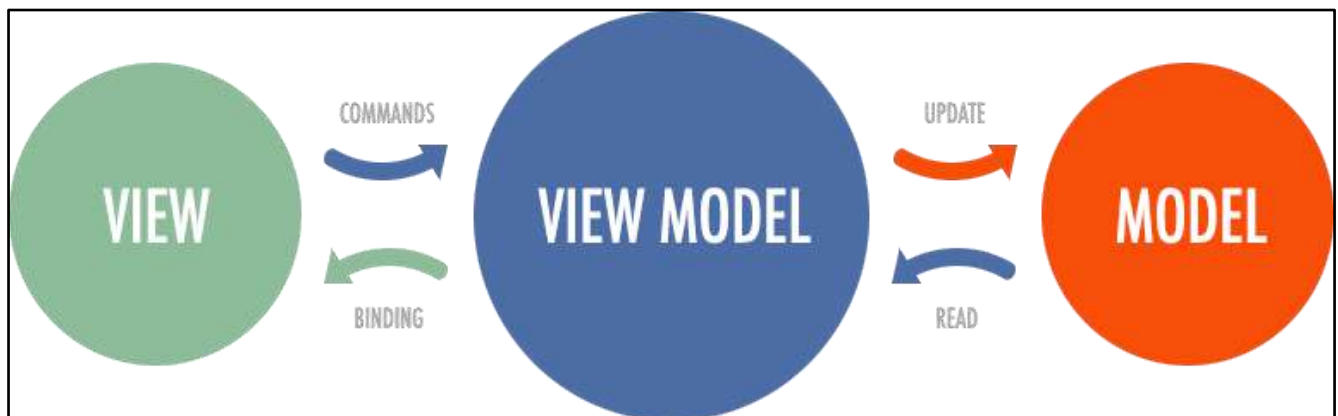
Η αρχιτεκτονική MVVM (Model-View-ViewModel) χρησιμοποιείται για τη δημιουργία διαχωρισμένων και επεκτάσιμων εφαρμογών, κυρίως σε περιβάλλοντα με πλούσιο γραφικό περιβάλλον όπως το WPF.

Η MVVM αρχιτεκτονική διαχωρίζει την εφαρμογή σε τρία κύρια μέρη: το Model, το View και το ViewModel. Το Model περιλαμβάνει τις κλάσεις και τις οντότητες (entities) που αντιπροσωπεύουν τα δεδομένα της εφαρμογής καθώς και τις εντολές CRUD (Create, Read, Update, Delete) για τη διαχείριση αυτών των δεδομένων.

Το View είναι υπεύθυνο για τη διεπαφή χρήστη, περιλαμβάνοντας κουμπιά, κείμενα και διαγράμματα, και αλληλοεπιδρά με το View Model μέσω εντολών και διπλής κατεύθυνσης δέσμευσης δεδομένων (two-way data binding).

Το View Model λειτουργεί ως γέφυρα μεταξύ του View και του Model, υλοποιώντας το πρότυπο σχεδίασης, τις εντολές και τη λογική αλληλεπίδρασης και διαχειρίζεται τα συμβάντα αλλαγής του μοντέλου. Μέσω αυτής της αρχιτεκτονικής, επιτυγχάνεται μια καθαρή διαίρεση της εφαρμογής, διευκολύνοντας τη συντήρηση και την επεκτασιμότητα της εφαρμογής, ενώ επιτρέπει την επαναχρησιμοποίηση κώδικα και τη βελτίωση της απόδοσης και της διαχείρισης του κώδικα.

Παρατίθεται γραφική απεικόνιση του μοντέλου MVVM.

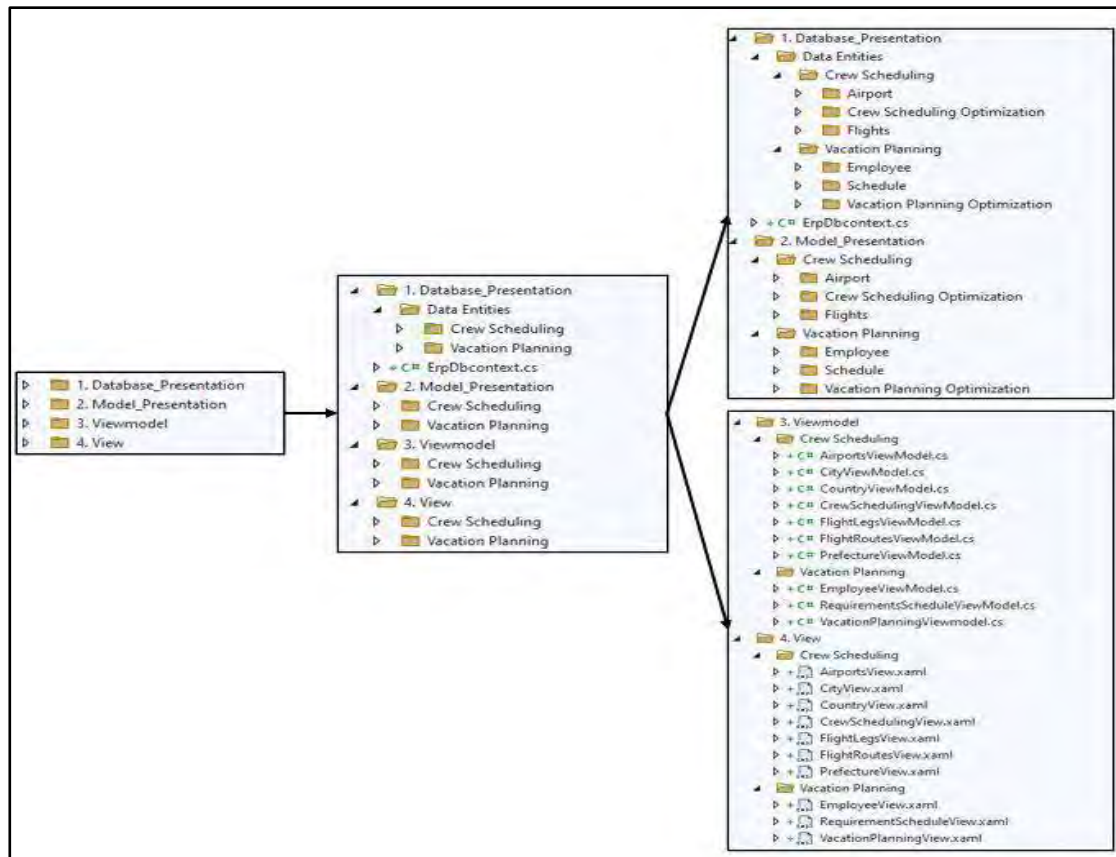


Εικόνα 7.1: Γραφική απεικόνιση MVVM αρχιτεκτονικής

7.2 Αρχιτεκτονική, MVVM Υλοποίηση.

7.2.1) Οργάνωση προγραμματιστικού περιβάλλοντος.

Παρατίθεται εικόνα (7.2) με παρουσίασης της οργάνωσης του προγραμματιστικού περιβάλλοντος.



Εικόνα 7.2: Οργάνωση εφαρμογής με βάση την MVVM αρχιτεκτονική

7.2.2) Βάση Δεδομένων

Στην ενότητα 7.2.2.1 παρουσιάζεται θεωρία κατασκευής πίνακα και σύνδεση της βάσης με την εφαρμογή. Στην ενότητα 7.2.2.2 παρουσιάζεται η βάση δεδομένων ως σχεσιακό διάγραμμα (Relational Database Diagram).

Στην ενότητα 7.2.2.3 παρουσιάζεται η βάση δεδομένων σε μορφή entities.

7.2.2.1) Βάση δεδομένων: Θεωρία

Για την κατασκευή ενός πίνακα εκτελούμε την εντολή Create Table στον SQL server. Στην συνέχεια κατασκευάζουμε την αντίστοιχη entity class του πίνακα σε c#.

Στην εικόνα (7.3) παρουσιάζεται παράδειγμα εντολής sql Create για τον πίνακα VPInput με Foreign key στο ReqScheduleInfo.

Στην εικόνα (7.4) παρουσιάζεται η αντίστοιχη κλάση entity για τον πίνακα VP Input.

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

SQL CREATE TABLE, VPInput

```
CREATE TABLE VPInput (
    VPID INT IDENTITY(1,1) PRIMARY KEY,
    VPCODE VARCHAR(50),
    VPDESCR VARCHAR(50),
    EMPLOYEETYPE VARCHAR(50),
    VPLOGICTYPE VARCHAR(50),
    MaxSatisfiedBids INT,
    SeparValue INT,
    IsDeleted BIT,
    SCHEDULEID INT,
    CONSTRAINT FK_SCHEDULEID FOREIGN KEY (SCHEDULEID)
        REFERENCES ReqScheduleInfo(SCHEDULEID)
);
```

Εικόνα 7.3: SQL Create table, example

Entity class, VPInput

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Erp.DataBasePresentation
{
    public class VPInputDataEntity
    {
        [Key]
        public int VPID { get; set; }

        [Column(TypeName = "varchar(50)")]
        public string VPCODE { get; set; }

        [Column(TypeName = "varchar(50)")]
        public string VPDESCR { get; set; }

        [Column(TypeName = "varchar(50)")]
        public string EMPLOYEETYPE { get; set; }

        [Column(TypeName = "varchar(50)")]
        public string VPLOGICTYPE { get; set; }

        [Column(TypeName = "int")]
        public int? MaxSatisfiedBids { get; set; }

        [Column(TypeName = "int")]
        public int? SeparValue { get; set; }

        [Column(TypeName = "bit")]
        public bool? IsDeleted { get; set; }

        public int? SCHEDULEID { get; set; }

        [ForeignKey("SCHEDULEID")]
        public virtual ReqScheduleInfoDataEntity Schedule
        {
        }
    }
}
```

Εικόνα 7.4: Entity, example

Στην εικόνα (7.5) παρουσιάζεται παράδειγμα εντολής SQL Select για τον πίνακα VPInput με Foreign Key τον πίνακα ReqScheduleInfo. Με τη χρήση του where statement, καλούνται μόνο τις γραμμές που δεν θεωρούνται διαγραμμένες.

```
SELECT V.VPID,V.VPCODE,V.VPDESCR,V.EMPLOYEETYPE,V.VPLOGICTYPE,
       V.SeparValue,V.MaxSatisfiedBids,V.ISDELETED,
       R.ID,R.ReqCode,R.ReqDescr,R.DateFrom,R.DateTo,R.LimitLineFixed
FROM VPInput as V
INNER JOIN ReqScheduleInfo AS R ON R.ID = V.SCHEDULEID
Where V.IsDeleted = 0
```

Εικόνα 7.5: SQL Select Table, example

Στην εικόνα (7.7) βλέπετε την οργάνωση της βάσης δεδομένων σε Data Entities στο προγραμματιστικό περιβάλλον της εφαρμογής.

Στην εικόνα (7.6) βλέπετε την κλάση Repository Base. Εισάγουμε το connection string. Έπειτα το χρησιμοποιούμε στο ERBDBContext.

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στην εικόνα (7.8) βλέπετε την κλάση ERPDBContext. Αποτελεί τη βασική κλάση οργάνωσης της βάσης δεδομένων. Εμπεριέχει την δήλωση των entities με το κανονικό όνομα των πινάκων όπως έχουν καταχωρηθεί στον SQL server , όπως και την εντολή σύνδεσης του προγράμματος με τη βάση.

```
public abstract class RepositoryBase
{
    private readonly string _connectionString;

    public RepositoryBase()
    {
        _connectionString = "Server=localhost\\SQLEXPRESS;
        Database=ERPDatabase;Encrypt=false;
        Integrated Security=true";
    }

    public DbContextOptions<ExpDbContext> options = new DbContextOptionsBuilder<ExpDbContext>()
        .UseSqlServer("Server=localhost\\SQLEXPRESS;Database=ERPDatabase;
        Trusted_Connection=True;MultipleActiveResultSets=True;Encrypt=false;")
        .Options;

    protected SqlConnection GetConnection()
    {
        return new SqlConnection(_connectionString);
    }
}
```

Repository Base:
Απλοϊκή κατασκευή
Connection String

Εικόνα 7.6: Repository Base

1. Database_Presentation

- Data Entities
 - Crew Scheduling
 - Airport
 - AirportsDataEntity.cs
 - CityDataEntity.cs
 - CountryDataEntity.cs
 - PrefectureEntity.cs
 - Crew Scheduling Optimization
 - CSInputDataEntity.cs
 - Flights
 - FlightLegsDataEntity.cs
 - FlightRoutesDataEntity.cs
 - Vacation Planning
 - Employee
 - EmployeeDataEntity.cs
 - LeaveBidsDataEntity.cs
 - LeaveStatusDataEntity.cs
 - Schedule
 - ReqScheduleInfoDataEntity.cs
 - ReqScheduleRowsDataEntity.cs
 - Vacation Planning Optimization
 - VPInputDataEntity.cs
- + C# ExpDbContext.cs

Εικόνα 7.7: Οργάνωση entities

```
public class ExpDbContext : Microsoft.EntityFrameworkCore.DbContext
{
    #region Database
    public DbSet<EmployeeDataEntity> Employees { get; set; }
    public DbSet<LeaveBidsDataEntity> LeaveBids { get; set; }
    public DbSet<LeaveStatusDataEntity> LeaveStatus { get; set; }
    public DbSet<ReqScheduleInfoDataEntity> ReqScheduleInfo { get; set; }
    public DbSet<ReqScheduleRowsDataEntity> ReqScheduleRows { get; set; }
    public DbSet<VPInputDataEntity> VPInput { get; set; }
    public DbSet<AirportsDataEntity> Airports { get; set; }
    public DbSet<CSInputDataEntity> CSInput { get; set; }
    public DbSet<FlightRoutesDataEntity> FlightRoutes { get; set; }
    public DbSet<FlightLegsDataEntity> FlightLegs { get; set; }

    #endregion

    #region Database Related Commands
    public static DbContextOptions<ExpDbContext> DbOptions { get; set; }
    public ExpDbContext(DbContextOptions<ExpDbContext> options)
        : base(options ?? throw new ArgumentNullException(nameof(options)),
        "DbContextOptions must be initialized before creating a " +
        "new ExpDbContext instance.")
    {
    }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<ItemDataEntity>().HasKey(i => i.ItemId);
        modelBuilder.Entity<Log>().HasKey(l => l.Id);
    }
    #endregion
}
```

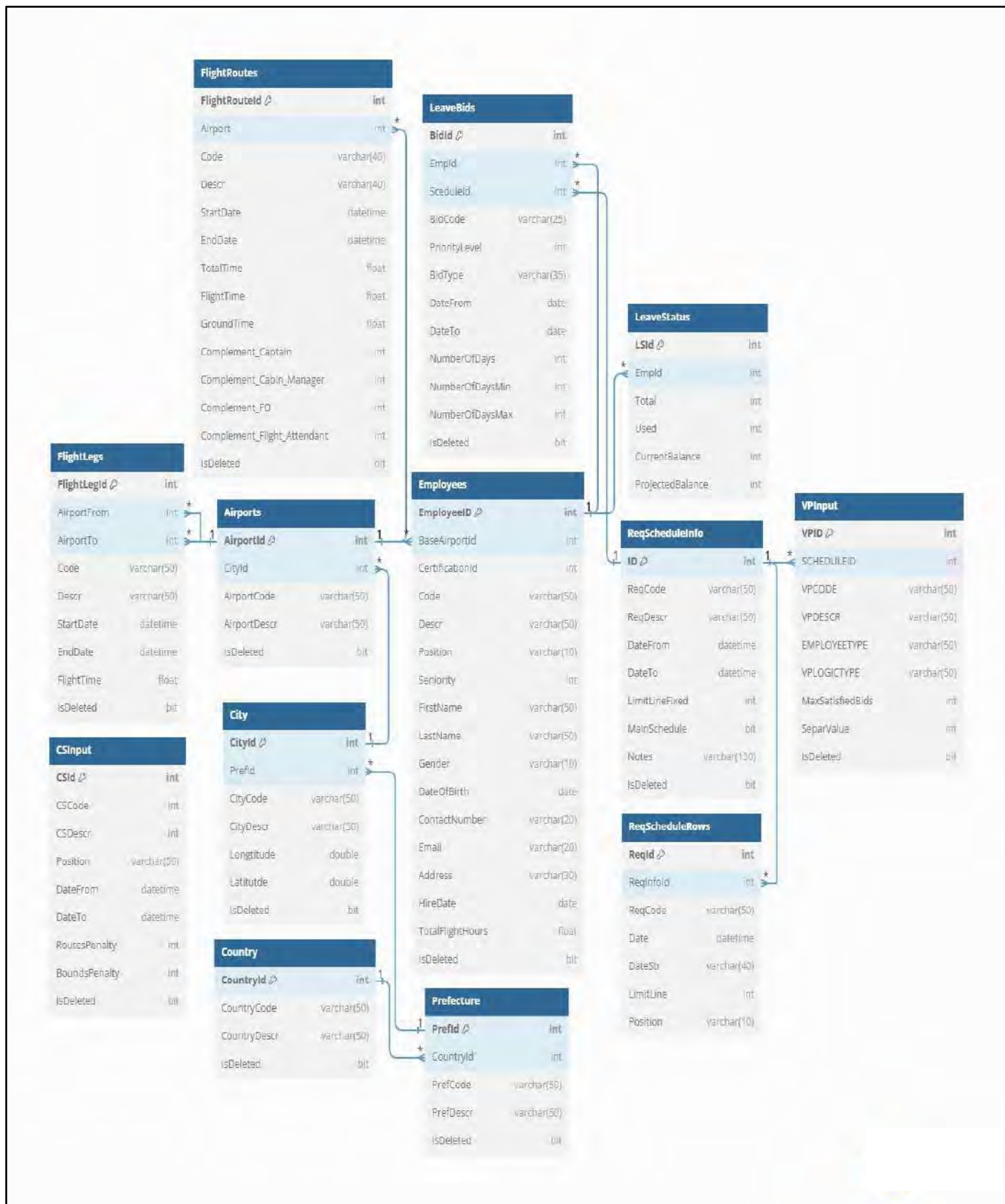
Δήλωση των Data Entities.

Βοηθητικές εντολές διαχείρισης της βάσης

Εικόνα 7.8: ExpDbContext

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.2.2.2) Βάση Δεδομένων: Σχεσιακό διάγραμμα



Εικόνα 7.9: Relational Database Diagram

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.2.2.3) Βάση Δεδομένων: Entities

<pre>public class EmployeeDataEntity { [Key] public int EmployeeID { get; set; } [Column(TypeName = "varchar(50)")] public string Code { get; set; } [Column(TypeName = "varchar(50)")] public string Descr { get; set; } [Column(TypeName = "varchar(50)")] public string FirstName { get; set; } [Column(TypeName = "varchar(50)")] public string LastName { get; set; } [Column(TypeName = "varchar(10)")] public string Gender { get; set; } [Column(TypeName = "varchar(30)")] public string Address { get; set; } [Column(TypeName = "datetime")] public DateTime? DateOfBirth { get; set; } [Column(TypeName = "varchar(20)")] public string ContactNumber { get; set; } [Column(TypeName = "varchar(20)")] public string Email { get; set; } [Column(TypeName = "varchar(30)")] public string Position { get; set; } [Column(TypeName = "float")] public float? TotalFlightHours { get; set; } }</pre>	<pre>public class LeaveBidsDataEntity { [Key] public int BidId { get; set; } public int? EmpId { get; set; } public int? SceduleId { get; set; } [Column(TypeName = "varchar(25)")] public string BidCode { get; set; } [Column(TypeName = "int")] public int? PriorityLevel { get; set; } [Column(TypeName = "varchar(35)")] public string BidType { get; set; } [Column(TypeName = "datetime")] public DateTime? DateFrom { get; set; } [Column(TypeName = "datetime")] public DateTime? DateTo { get; set; } [Column(TypeName = "varchar(35)")] public string DateFromStr { get; set; } [Column(TypeName = "varchar(35)")] public string DateToStr { get; set; } [Column(TypeName = "int")] public int? NumberOfDays { get; set; } [Column(TypeName = "int")] public int? NumberOfDaysMin { get; set; } [Column(TypeName = "int")] public int? NumberOfDaysMax { get; set; } [Column(TypeName = "bool")] public bool IsDeleted { get; set; } }</pre>	<pre>public class LeaveStatusDataEntity { [Key] public int LSId { get; set; } [Column(TypeName = "int")] public int? Total { get; set; } [Column(TypeName = "int")] public int? Used { get; set; } [Column(TypeName = "int")] public int? CurrentBalance { get; set; } [Column(TypeName = "int")] public int? ProjectedBalance { get; set; } public int? EmpId { get; set; } [ForeignKey("EmpId")] public virtual EmployeeDataEntity Employee { get; set; } }</pre>
---	--	---

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

<pre>[Column(TypeName = "datetime")] public DateTime? HireDate { get; set; } [Column(TypeName = "int")] public int? Seniority { get; set; } [Column(TypeName = "bool")] public bool IsDeleted { get; set; } public int? BaseAirportId { get; set; } [Column(TypeName = "int")] public int? LowerBound { get; set; } [Column(TypeName = "int")] public int? UpperBound { get; set; } [ForeignKey("BaseAirportId")] public virtual AirportsDataEntity Airports { get; set; } }</pre>	<pre>[ForeignKey("EmpId")] public virtual EmployeeDataEntity Employees { get; set; } [ForeignKey("SceduleId")] public virtual ReqScheduleInfoDataEntity ReqSchedule { get; set; } }</pre>	
<pre>public class ReqScheduleInfoDataEntity { [Key] public int ID { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string ReqCode { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string ReqDescr { get; set; } [Required] [Column(TypeName = "varchar(150)")] public string Notes { get; set; } [Column(TypeName = "datetime")] public DateTime? DateFrom { get; set; } [Column(TypeName = "datetime")] public DateTime? DateTo { get; set; } }</pre>	<pre>public class ReqScheduleRowsDataEntity { [Key] public int ReqId { get; set; } [Column(TypeName = "varchar(50)")] public string ReqCode { get; set; } [Column(TypeName = "varchar(30)")] public string Position { get; set; } [Column(TypeName = "int")] public int? LimitLine { get; set; } [Column(TypeName = "varchar(40)")] public string DateStr { get; set; } [Column(TypeName = "datetime")] public DateTime? Date { get; set; } }</pre>	<pre>public class VPInputDataEntity { [Key] public int VPID { get; set; } [Column(TypeName = "varchar(50)")] public string VPCODE { get; set; } [Column(TypeName = "varchar(50)")] public string VPDESCR { get; set; } [Column(TypeName = "varchar(50)")] public string EMPLOYEE TYPE { get; set; } [Column(TypeName = "varchar(50)")] public string VPLOGICTYPE { get; set; } [Column(TypeName = "int")] public int? MaxSatisfiedBids { get; set; } [Column(TypeName = "int")]</pre>

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

<pre>[Column(TypeName = "int")] public int? LimitLineFixed { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } [Column(TypeName = "bit")] public bool? MainSchedule { get; set; } }</pre>		<pre>public int? SeparValue { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } public int? SCHEDULEID { get; set; } [ForeignKey("SCHEDULEID")] public virtual ReqScheduleInfoDataEntity Schedule { get; set; } }</pre>
<pre>public class CountryDataEntity { [Key] public int CountryId { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string CountryCode { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string CountryDescr { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } }</pre>	<pre>public class PrefectureDataEntity { [Key] public int PrefId { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string PrefCode { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string PrefDescr { get; set; } public int CountryId { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } [ForeignKey("CountryId")] public virtual CountryDataEntity Country { get; set; } }</pre>	<pre>public class CityDataEntity { [Key] public int CityId { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string CityCode { get; set; } [Required] [Column(TypeName = "varchar(50)")] public string CityDescr { get; set; } [Column(TypeName = "double")] public double? Longitude { get; set; } [Column(TypeName = "double")] public double? Latitude { get; set; } [Column(TypeName = "int")] public int? Population { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } public int PrefId { get; set; } [ForeignKey("PrefId")] public virtual PrefectureEntity Prefecture { get; set; } }</pre>
<pre>public class AirportsDataEntity { [Key]</pre>	<pre>public class FlightLegsDataEntity { [Key]</pre>	<pre>public class FlightRoutesDataEntity { [Key]</pre>

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

<pre> public int AirportID { get; set; } [Column(TypeName = "varchar(50)")] public string AirportCode { get; set; } [Column(TypeName = "varchar(50)")] public string AirportDescr { get; set; } [Column(TypeName = "bool")] public bool IsDeleted { get; set; } public int? CityId { get; set; } [ForeignKey("CityId")] public virtual CityDataEntity City { get; set; } } </pre>	<pre> public int FlightLegId { get; set; } [Column(TypeName = "varchar(40)")] public string Code { get; set; } [Column(TypeName = "varchar(40)")] public string Descr { get; set; } [Column(TypeName = "datetime")] public DateTime? StartDate { get; set; } [Column(TypeName = "datetime")] public DateTime? EndDate { get; set; } [Column(TypeName = "float")] public float? FlightTime { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } public int AirportFrom { get; set; } public int AirportTo { get; set; } [ForeignKey("AirportFrom")] public virtual AirportsDataEntity Airport1 { get; set; } [ForeignKey("AirportTo")] public virtual AirportsDataEntity Airport2 { get; set; } } </pre>	<pre> public int FlightRouteId { get; set; } [Column(TypeName = "varchar(40)")] public string Code { get; set; } [Column(TypeName = "varchar(40)")] public string Descr { get; set; } [Column(TypeName = "datetime")] public DateTime? StartDate { get; set; } [Column(TypeName = "datetime")] public DateTime? EndDate { get; set; } [Column(TypeName = "float")] public float? FlightTime { get; set; } [Column(TypeName = "float")] public float? GroundTime { get; set; } [Column(TypeName = "float")] public float? TotalTime { get; set; } [Column(TypeName = "int")] public int? Complement_Captain { get; set; } [Column(TypeName = "int")] public int? Complement_FO { get; set; } [Column(TypeName = "int")] public int? Complement_Cabin_Manager { get; set; } [Column(TypeName = "int")] public int? Complement_Flight_Attendant { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } public int AirportId { get; set; } [ForeignKey("AirportId")] public virtual </pre>
---	--	--

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

		AirportsDataEntity Airport { get; set; } }
	<pre> public class CSInputDataEntity { [Key] public int CSID { get; set; } [Column(TypeName = "varchar(50)")] public string CSCODE { get; set; } [Column(TypeName = "varchar(30)")] public string CSDESCR { get; set; } [Column(TypeName = "varchar(30)")] public string EMPLOYEE TYPE { get; set; } [Column(TypeName = "datetime")] public DateTime? DateFrom { get; set; } [Column(TypeName = "datetime")] public DateTime? DateTo { get; set; } [Column(TypeName = "int")] public int? RoutesPenalty { get; set; } [Column(TypeName = "int")] public int? BoundsPenalty { get; set; } [Column(TypeName = "bit")] public bool? IsDeleted { get; set; } } </pre>	

7.2.3) Model

Στην ενότητα 7.2.3.1 παρουσιάζεται η θεωρία κατασκευής μίας κλάσης, η κλήση της στο View Model και η σύνδεση (Binding) των μεταβλητών της στο View.

Στην ενότητα 7.2.3.2 παρουσιάζονται όλες οι Model κλάσεις της εφαρμογής. Οι VacationPlanningInputData, VacationPlanningOutputData και υπόλοιπες της βελτιστοποίησης ανάθεσης αδειών δεν παρουσιάζονται σε αυτή την ενότητα. Παρουσιάζονται εκτενώς στην ενότητα 7.4.1.

7.2.3.1) Model: Θεωρία

Κατασκευή Model class

Στην εικόνα (7.11) παρουσιάζεται η απλοποιημένη κλάση VacationPlanningInputData.

Στην εικόνα (7.10) παρουσιάζεται ο πραγματικός τρόπος δήλωσης μεταβλητών μιας κλάσης στη εφαρμογή με την λειτουργία OnPropertyChanged.

Με την λειτουργία OnPropertyChanged, οι μεταβλητές μας ενημερώνεται από τις αλλαγές που γίνονται από το View ή μέσω Get εντολών από την βάση δεδομένων.

```
public class VacationPlanningInputData : RecordBaseModel
{
    #region General
    [References]
    public int VPID { get; set; }
    [References]
    public string VPCode { get; set; }
    [References]
    public string VPDescr { get; set; }
    [References]
    public BasicEnums.EmployeeType EmployeeType { get; set; }
    [References]
    public BasicEnums.VPLogicType VPLogicType { get; set; }
    [References]
    public int MaxSatisfiedBids { get; set; }
    [References]
    public int SeparValue { get; set; }
    [References]
    public ReqScheduleInfoData Schedule { get; set; }
    [References]
    public ObservableCollection<EmployeeData> Employees { get; set; }
    [References]
    public DateTime[] Dates { get; set; }
    [References]
    public string[] DatesStr { get; set; }
    #endregion
}
```

Εικόνα 7.11: Model class

```
private string _VPCode;

19 references
public string VPCode
{
    get { return _VPCode; }
    set { _VPCode = value; OnPropertyChanged("VPCode"); }
}
```

Εικόνα 7.10: Δήλωση μεταβλητών, Property Changed

Αρχικοποίηση, δήλωση Model class στο View model

```
public class VacationsPlanningViewModel : ViewModelBase
{
    #region Δήλωση κλάσης
    private VacationPlanningInputData inputdata;
    [References]
    public VacationPlanningInputData InputData
    {
        get { return inputdata; }
        set
        {
            inputdata = value;
            NotifyPropertyChanged(nameof(InputData));
        }
    }
    #endregion

    #region Αρχικοποίηση κλάσης στην έναρξη του Viewmodel
    public VacationsPlanningViewModel()
    {
        InputData = new VacationPlanningInputData();
        InputData.Schedule = new ReqScheduleInfoData();
        InputData.Schedule.RegScheduleRowsData = new ObservableCollection<ReqScheduleRowsData>();
    }
    #endregion
}
```

Εικόνα 7.12: Αρχικοποίηση και Δήλωση Model class

Σύνδεση μεταβλητών της κλάσης στο View

```
<TextBlock Text="Max Satisfied Bids (Bmax): "
            Grid.Row="3" Grid.Column="0" />
<TextBlock Text="Separation Value (Se): "
            Grid.Row="3" Grid.Column="2" />
<TextBox Text="{Binding InputData.Bmax, Mode=TwoWay}"
          Grid.Row="3" Grid.Column="1" />
<TextBox Text="{Binding InputData.Se, Mode=TwoWay}"
          Grid.Row="3" Grid.Column="3" />
```

Εικόνα 7.13: Σύνδεση μεταβλητών με το View

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.2.3.2) Model: Κλάσεις

Στον παρακάτω πίνακα παρουσιάζονται οι model κλάσεις του προγράμματος.

<pre> public class EmployeeData { public int EmployeeId { get; set; } public string Code { get; set; } public BasicEnums.EmployeeType Position { get; set; } public int Seniority { get; set; } public AirportData BaseAirport { get; set; } public ReqScheduleInfoData MainSchedule { get; set; } public LeaveStatusData LeaveStatus { get; set; } public LeaveBidsData LeaveBidInfo { get; set; } public ObservableCollection<LeaveBidsDa taStatic> LeaveBidDataGridStatic { get;set; } #region Γενικά στοιχεία υπαλλήλου ... #endregion } </pre>	<pre> public class LeaveStatusData { public EmployeeData Employee { get; set; } public int Total { get; set; } public int Used { get; set; } public int CurrentBalance { get { return Total - Used; } } } </pre>	<pre> public class LeaveBidsData { public EmployeeData Employee { get; set; } public ReqScheduleInfoData Schedule { get; set; } public int BidId { get; set; } public string BidCode { get; set; } public int PriorityLevel { get; set; } public BasicEnums.BidType BidType { get; set; } public DateTime DateFrom { get; set; } public DateTime DateTo { get; set; } public int NumberOfDays { get; set; } public int NumberOfDaysMin { get; set; } public int NumberOfDaysMax { get; set; } } </pre>
<pre> public class ReqScheduleInfoData { public int ID { get; set; } public string ReqCode { get; set; } public DateTime DateFrom { get; set; } public DateTime DateTo { get; set; } public BasicEnums.EmployeeType Position { get; set; } public bool MainSchedule { get; set; } public int LimitLineFixed { get; set; } public ObservableCollection<ReqSchedule RowsData> ReqScheduleRowsData { get; set; } } </pre>	<pre> public class ReqScheduleRowsData { public int ReqId { get; set; } public string ReqCode { get; set; } public BasicEnums.EmployeeTy pe Position { get; set; } public DateTime Date { get; set; } public int LimitLine { get; set; } } </pre>	<pre> public class CountryData { public int CountryId { get; set; } public string CountryCode { get; set; } public string CountryDescr { get; set; } } </pre>
<pre> public class PrefectureData { public int PrefId { get; set; } public string PrefCode { get; set; } public string PrefDescr { get; set; } public CountryData Country { </pre>	<pre> public class CityData { public int CityId { get; set; } public string CityCode { get; set; } public string CityDescr { get; set; } </pre>	<pre> public class FlightLegsData { public int FlightLegId { get; set; } public string Code { get; set; } public string Descr { get; set; } public AirportData </pre>

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

<pre>get; set; } }</pre>	<pre>} public PrefectureData Prefecture { get; set; } }</pre>	<pre>AirportDataFrom { get; set; } public AirportData AirportDataTo { get; set; } public float FlightTime { get; set; } public DateTime StartDate { get; set; } public DateTime EndDate { get; set; } }</pre>
<pre>public class FlightLegsData { public int FlightLegId { get; set; } public string Code { get; set; } public string Descr { get; set; } public AirportData AirportDataFrom { get; set; } public AirportData AirportDataTo { get; set; } public float FlightTime { get; set; } public DateTime StartDate { get; set; } public DateTime EndDate { get; set; } }</pre>	<pre>public class FlightRoutesData { public int FlightRouteId { get; set; } public string Code { get; set; } public string Descr { get; set; } public AirportData Airport { get; set; } public DateTime StartDate { get; set; } public DateTime EndDate { get; set; } public float TotalTime { get; set; } public float FlightTime { get; set; } public float GroundTime { get; set; } public int Complement_Captain { get; set; } public int Complement_FO { get; set; } public int Complement_Cabin_Mana ger { get; set; } public int Complement_Flight_Att endant { get; set; } }</pre>	<pre>public class CSInputData { public int Id { get; set; } public string Code { get; set; } public string Descr { get; set; } public DateTime DateFrom { get; set; } public DateTime DateTo { get; set; } public BasicEnums.EmployeeType Position { get; set; } public int RoutesPenalty { get; set; } public int BoundsPenalty { get; set; } public ObservableCollection<FlightRou tesData> FlightRoutesData { get; set; } public ObservableCollection<EmployeeD ata> Employees { get; set; } }</pre>

7.2.4) Εντολές διαχείρισης δεδομένων / CRUD Commands

7.2.4.1) Get Commands

Στην ενότητα 7.2.4.1 παρουσιάζονται 2 παραδείγματα εντολών Get για την κλάση VPIInputData. Στους παρακάτω πίνακες παρουσιάζονται οι εντολές GetVPIInputData, GetVPIInputChooserData. Η εντολή GetVPIInputData καλεί όλα τα πείραμα ανάθεσης αδειών από τη βάση. Η εντολή GetVPIInputChooserData καλεί ένα μεμονωμένο πείραμα με βάση τον κωδικό ή id του πειράματος που θέλουμε να καλέσουμε.

Επεξήγηση δομής των εντολών Get

GetVPIInputData	GetVPIInputChooserData
Περιγραφή: Η εντολή Get δέχεται ως δεδομένο εισαγωγής την δυαδική μεταβλητή ShowDeleted Από το κουμπί [Show Deleted] που είδαμε στο εγχειρίδιο της εφαρμογής. Η εντολή ενεργοποιείται για παράδειγμα πατώντας ένα οποιοδήποτε κουμπί [f] και γεμίζει δεδομένα στο Data Grid	Περιγραφή : Η εντολή GetChooser δέχεται ως δεδομένο εισαγωγής το κωδικό ή το id και επιστρέφει την επιλεγμένη γραμμή από τη βάση δεδομένων .Η εντολή Get ενεργοποιείται για παράδειγμα από το κουμπί [refresh] που είδαμε στο εγχειρίδιο εφαρμογής
1) Εισαγωγή SQL Query .Η τιμή του Show Deleted εισάγεται και προστίθεται στο Where	1)Επιλογή γραμμής με βάση το Id ή το Code
2) Αρχικοποίηση ObservableCollection όπου θα εισαχθούν όλες οι γραμμές από τη βάση	2) Μεταφορά δεδομένων στην γραμμή της Model κλάσης
3)Μεταφορά δεδομένων στην γραμμή της Model κλάσης	3)Επιστροφή της γραμμής / Model class στο View model
4) Δεδομένα σχετικά με το πείραμα του Vacation Planning	
5)Δεδομένα σχετικά με το Schedule .Τα δεδομένα αυτά εισάγονται μέσω Foreign Key	
6) α. Αρχικοποίηση γραμμής β. Προσθήκη γραμμής στο Observable Collection	

Get Command

```

public ObservableCollection<VacationPlanningInputData> GetVPInputData(bool ShowDeleted)
{
    ObservableCollection<VacationPlanningInputData> DataGrid= new
    ObservableCollection<VacationPlanningInputData>();
    string FilterStr = "";

    using (var connection = GetConnection())
    using (var command = new SqlCommand())
    {
        connection.Open();
        command.Connection = connection;

        if (ShowDeleted == false)
        {
            command.Parameters.AddWithValue("@ShowDeleted", ShowDeleted);
            FilterStr = String.Format(@"and V.IsDeleted = @ShowDeleted");
        }
        string SQLQuery = @"SELECT
V.VPID,V.VPCODE,V.VPDESCR,V.EMPLOYEEYPE,V.VPLOGICTYPE,V.SeparValue,
V.MaxSatisfiedBids,V.ISDELETED,
R.ID,R.ReqCode,R.ReqDescr,R.DateFrom,R.DateTo,R.LimitLineFixed
FROM VPInput as V
INNER JOIN ReqScheduleInfo AS R ON R.ID = V.SCHEDULEID
Where 1=1 {0}";

        command.CommandText = string.Format(SQLQuery, FilterStr);

        using (var reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                #region Initialize Row
                VacationPlanningInputData data = new VacationPlanningInputData();
                data.Schedule = new ReqScheduleInfoData();
                #endregion

                #region Populate Data Row

                #region Populate VP
                data.VPId = int.Parse(reader["VPID"].ToString());
                data.VPCode = reader["VPCODE"].ToString();
                data.VPDescr = reader["VPDESCR"].ToString();
                data.Bmax = int.Parse(reader["MaxSatisfiedBids"].ToString());
                data.Se = int.Parse(reader["SeparValue"].ToString());
                data.EmployeeType =
                (BasicEnums.EmployeeType)Enum.Parse(typeof(BasicEnums.EmployeeType),
                reader["EMPLOYEEYPE"].ToString());
                data.VPLogicType =
                (BasicEnums.VPLogicType)Enum.Parse(typeof(BasicEnums.VPLogicType),
                reader["VPLogictype"].ToString());
                data.IsDeleted = bool.Parse(reader["ISDELETED"].ToString());
                #endregion

                #region Populate Schedule

                data.Schedule.ID = int.Parse(reader["ID"].ToString());
                data.Schedule.ReqCode = reader["ReqCode"].ToString();
                data.Schedule.ReqDescr = reader["ReqDescr"].ToString();
                data.Schedule.DateFrom = DateTime.Parse(reader["DateFrom"].ToString());
                data.Schedule.DateTo = DateTime.Parse(reader["DateTo"].ToString());
                data.Schedule.LimitLineFixed = int.Parse(reader["LimitLineFixed"].ToString());
                data.Schedule.DateFromStr = data.Schedule.DateFrom.ToString("dd/MM/yyyy");
                data.Schedule.DateToStr = data.Schedule.DateTo.ToString("dd/MM/yyyy");
                #endregion

                //Add Row to ObservableCollection
                DataGrid.Add(data);
            }
        }
        connection.Close();
    }
    return DataGrid;
}

```

Get Chooser Command

```

public VacationPlanningInputData GetVPInputChooserData(VacationPlanningInputData Data)
{
    string FilterStr = "";
    using (var connection = GetConnection())
    using (var command = new SqlCommand())
    {
        connection.Open();
        command.Connection = connection;
        if (Data.VPId > 0)
        {
            command.Parameters.AddWithValue("@VPID", Data.VPId);
            FilterStr += " and V.VPID = @VPID";
        }
        else if (!string.IsNullOrEmpty(Data.VPCode))
        {
            command.Parameters.AddWithValue("@VPCODE", Data.VPCode);
            FilterStr += " and V.VPCODE = @VPCODE";
        }
        command.CommandText = string.Format("SELECT V.VPID, V.VPCODE, V.VPDESCR,
                                            V.EMPLOYEETYPE, V.VPLOGICTYPE,
                                            V.MaxSatisfiedBids, V.ISDELETED,
                                            V.SeparValue,
                                            R.ID, R.ReqCode,
                                            R.ReqDescr, R.DateFrom, R.DateTo
                                            FROM VPInput AS V
                                            INNER JOIN ReqScheduleInfo AS R ON R.ID =V.SCHEDULEID
                                            WHERE 1=1 {0}", FilterStr);

        using (var reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                #region Populate Data

                #region Populate Vacation Planning Data

                Data.VPId = int.Parse(reader["VPID"].ToString());
                Data.VPCode = reader["VPCODE"].ToString();
                Data.VPDescr = reader["VPDESCR"].ToString();
                Data.EmployeeType = (BasicEnums.EmployeeType)Enum.Parse(typeof(BasicEnums.EmployeeType),
                                                                    reader["EMPLOYEETYPE"].ToString());
                Data.VPLogicType = (BasicEnums.VPLogicType)Enum.Parse(typeof(BasicEnums.VPLogicType),
                                                                    reader["VPLogicType"].ToString());
                Data.Bmax = int.Parse(reader["MaxSatisfiedBids"].ToString());
                Data.Se = int.Parse(reader["SeparValue"].ToString());
                Data.IsDeleted = bool.Parse(reader["ISDELETED"].ToString());

                #endregion

                #region Populate Schedule Data

                Data.Schedule = new ReqScheduleInfoData();

                Data.Schedule.ID = int.Parse(reader["ID"].ToString());
                Data.Schedule.ReqCode = reader["ReqCode"].ToString();
                Data.Schedule.ReqDescr = reader["ReqDescr"].ToString();
                Data.Schedule.DateFrom = DateTime.Parse(reader["DateFrom"].ToString());
                Data.Schedule.DateTo = DateTime.Parse(reader["DateTo"].ToString());
                Data.Schedule.DateFromStr = Data.Schedule.DateFrom.ToString(); //THELEI ALAGH
                Data.Schedule.DateToStr = Data.Schedule.DateTo.ToString();

                #endregion

                #endregion
            }
        }
        connection.Close();
    }
    return Data;
}

```

Diagram illustrating the flow of data in the `GetVPInputChooserData` method:

- 1**: The `command.CommandText` is constructed using the `FilterStr` and the `VPID` or `VPCODE` from the input data.
- 2**: The `reader` is used to populate the `Data` object with values from the database. This includes the `VPID`, `VPCODE`, `VPDESCR`, `EMPLOYEETYPE`, `VPLOGICTYPE`, `MaxSatisfiedBids`, `ISDELETED`, `SeparValue`, `ID`, `ReqCode`, `ReqDescr`, `DateFrom`, `DateTo`, `DateFromStr`, and `DateToStr`.
- 3**: The `Data` object is returned.

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.2.4.2) Add, Save Commands

Στην ενότητα 7.2.4.2 παρουσιάζονται 2 παραδείγματα εντολών Add και Save για την κλάση EmployeeData. Στους παρακάτω πίνακες παρουσιάζονται οι εντολές AddEmployeeData, SaveEmployeeData. Η εντολή AddEmployeeData προσθέτει ένα νέο υπάλληλο στην βάση, θέτοντας default τιμές. Η εντολή SaveEmployeeData αποθηκεύει αλλαγές σε ένα ήδη υπάρχον αποθηκευμένο υπάλληλο.

Επεξήγηση δομής της εντολής Add

AddEmployeeData
1) Έλεγχος ήδη υπάρχουσας γραμμής με τον ίδιο κωδικό στη βάση. Αν υπάρχει, η εντολή τερματίζεται και εμφανίζεται error μήνυμα
2) Αν δεν υπάρχει. Μεταφέρουμε το κωδικό και την περιγραφή από το Model Class στο Data Entity και θέτουμε default τιμές για τις υπόλοιπες μεταβλητές
3) Προσθήκη και αποθήκευση της γραμμής στο αντίστοιχο πίνακα της βάσης. Επιστροφή μιας int τιμής που λειτουργεί ως flag για εμφάνιση των αντίστοιχων message boxes στο view model.
4) Σε περίπτωση σφάλματος λόγω κώδικα το Exception αποθηκεύεται στη βάση μέσω του try – catch για λογούς debugging.

Επεξήγηση δομής της εντολής Save

SaveEmployeeData
1) Επιλογή αποθηκευμένης γραμμής από τη βάση δεδομένων μέσω του Id. Σε περίπτωση που δεν υπάρχει η εντολή τερματίζεται
2) Μεταφορά των δεδομένων από την Model Class στο Data Entity.
3) Αποθήκευση αλλαγών στη βάση δεδομένων και επιστροφή της τιμής int 1 για εμφάνιση των αντίστοιχων μηνυμάτων στο χρήστη

Add Command

```

public int AddEmployeeData(EmployeeData flatData)
{
    try
    {
        using (var dbContext = new ErpDbContext(options))
        {
            // Separate query from execution
            1 { var existingItemQuery = dbContext.Employees.Where(r => r.Code == flatData.Code);
              var existingItem = existingItemQuery.SingleOrDefault();
              // Execute the query and get the result

              if (existingItem == null)
              {
                  var newItem = new EmployeeDataEntity();
                  // Insert new item
                  2 { newItem.Code = flatData.Code;
                    newItem.Descr = flatData.Descr;
                    newItem.FirstName = " ";
                    newItem.LastName = " ";
                    newItem.Gender = BasicEnums.Gender.Male.ToString();
                    newItem.ContactNumber = " ";
                    newItem.Email = " ";
                    newItem.Address = " ";
                    newItem.Position = BasicEnums.EmployeeType.Captain.ToString();
                    newItem.TotalFlightHours = 0;
                    newItem.Seniority = 0;
                    newItem.Language = 0;
                    newItem.LowerBound = 70;
                    newItem.UpperBound = 80;
                    newItem.DateOfBirth = DateTime.Now;
                    newItem.HireDate = DateTime.Now;
                    newItem.IsDeleted = false;
                    newItem.BaseAirportId = dbContext.Airports.FirstOrDefault().AirportID;

                    dbContext.Employees.Add(newItem);
                    dbContext.SaveChanges();
                    3 return 0;
                  }
              }
              else
              {
                  return 1;
              }
            }
        }
    }
    catch (Exception ex)
    {
        LogError(ex, "AddEmployeeData", "Notes");
        4 return 2;
    }
}

```

Save Command

```

public int SaveEmployeeData(EmployeeData flatData)
{
    try
    {
        using (var dbContext = new ErpDbContext(options))
        {
            // Separate query from execution
            int employeeId = flatData.EmployeeId;
            var existingQuery = dbContext.Employees.Where(c => c.EmployeeID == employeeId);
            var existing = existingQuery.SingleOrDefault();
            if (existing != null)
            {
                // Update existing customer
                existing.Code = flatData.Code;
                existing.Descr = flatData.Descr;
                existing.FirstName = flatData.FirstName;
                existing.LastName = flatData.LastName;
                existing.Gender = flatData.Gender.ToString();
                existing.ContactNumber = flatData.ContactNumber;
                existing.Email = flatData.Email;
                existing.Address = flatData.Address;
                existing.Position = flatData.Position.ToString();
                existing.TotalFlightHours = flatData.TotalFlightHours;
                existing.Seniority = flatData.Seniority;
                existing.Language = flatData.Language; ;
                existing.DateOfBirth = DateTime.Now;
                existing.HireDate = DateTime.Now;
                existing.LowerBound = flatData.EmpCrSettings.LowerBound;
                existing.UpperBound = flatData.EmpCrSettings.UpperBound;
                existing.IsDeleted = flatData.IsDeleted;

                existing.BaseAirportId = flatData.BaseAirport.Id;

                dbContext.SaveChanges();
                return 1;
            }
            else
            {
                return -1;
            }
        }
    }
    catch (Exception ex)
    {
        LogError(ex, "SaveEmployeeData", "Notes");
        return -1;
    }
}

```

1

2

3

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.2.5) View Model

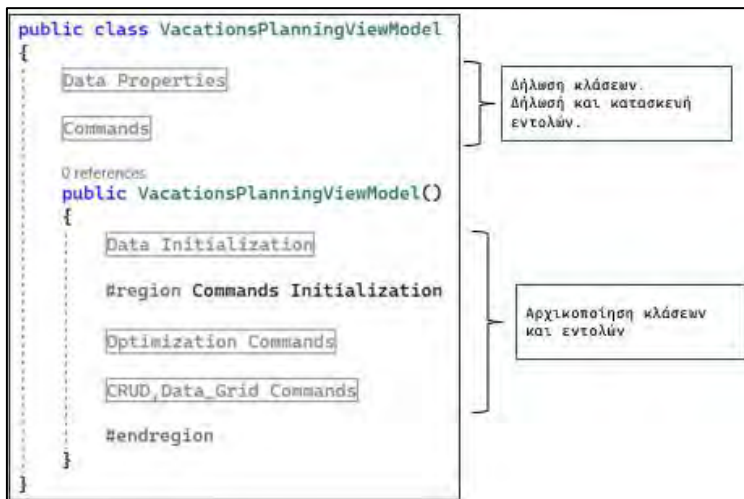
Στην εικόνα (7.15) παρουσιάζεται η οργάνωση των view models της εφαρμογής στο προγραμματιστικό περιβάλλον.

Για την παρουσίαση της δομής ενός view model επιλέγουμε ως παράδειγμα το VacationPlanningViewModel.

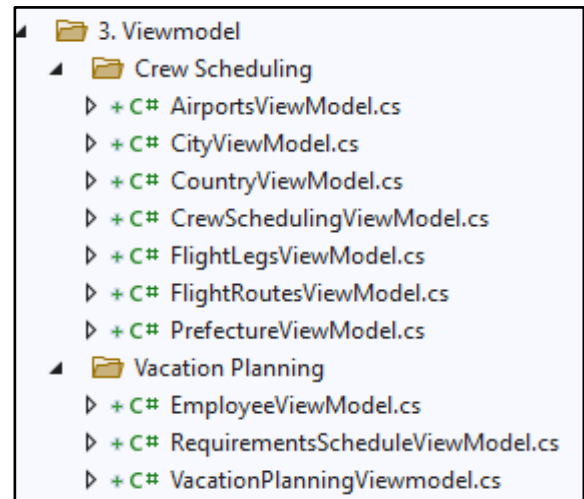
Στην εικόνα (7.14) παρουσιάζεται η οργάνωση της δομής ενός view model .

Στην εικόνα (7.16) παρουσιάζεται η αρχικοποίηση εντολών και κλάσεων του view model.

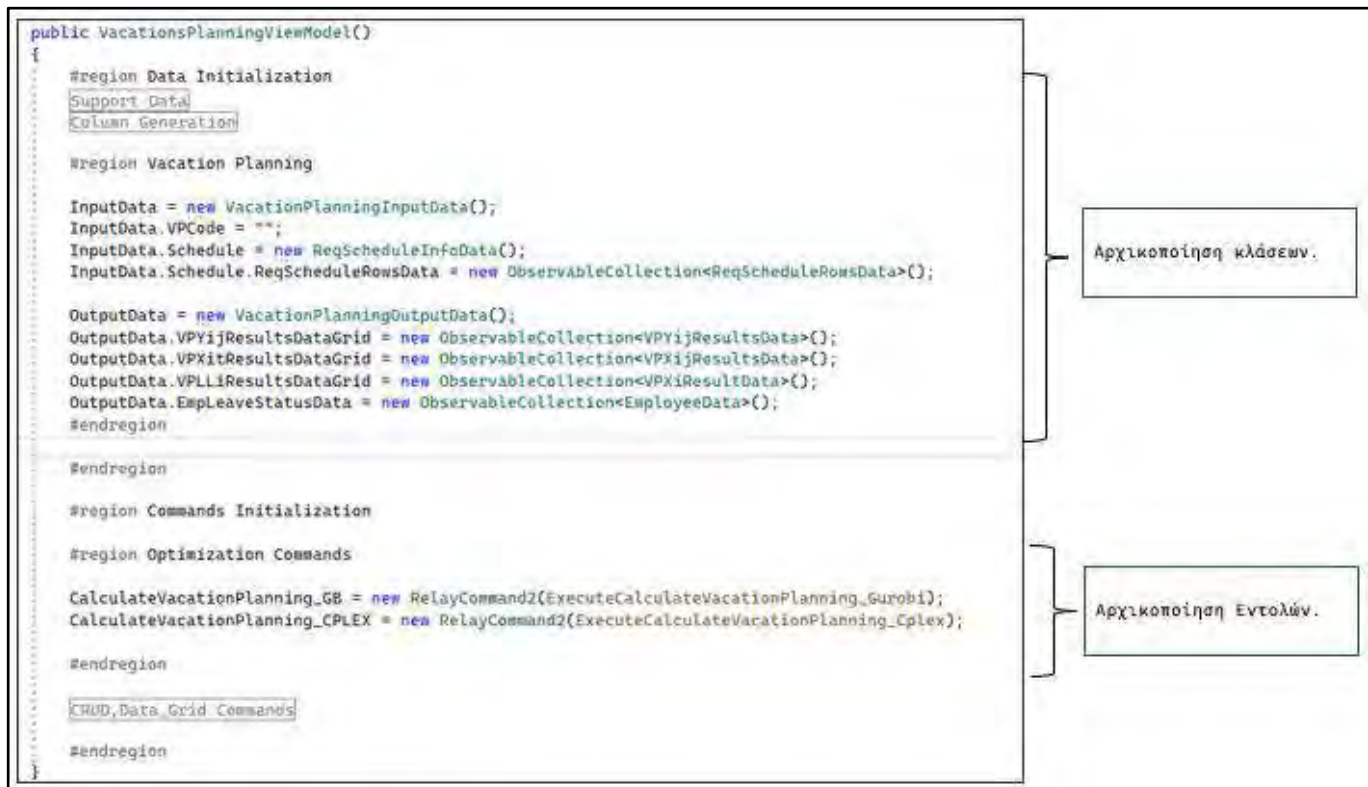
Στην εικόνα (7.17) παρουσιάζεται δήλωση κλάσεων και εντολών του view model.



Εικόνα 7.14: Δομή View model



Εικόνα 7.15: Φάκελος οργάνωσης View models



Εικόνα 7.16: Δομή View model 2

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```
#region Vacation Planning Data
private VacationPlanningInputData inputdata;
PS references
public VacationPlanningInputData InputData
{
    get { return inputdata; }
    set
    {
        inputdata = value;
        INotifyPropertyChanged(nameof(InputData));
    }
}

private VacationPlanningOutputData outputdata;
PS references
public VacationPlanningOutputData OutputData
#endregion
#endregion

#region Commands
#region CRUD Commands

#region Save

private ViewModelCommand _SaveInputCommand;
0 references
public ICommand SaveInputCommand
{
    get
    {
        if (_SaveInputCommand == null)
        {
            _SaveInputCommand = new ViewModelCommand(ExecuteSaveInputCommand);
        }
        return _SaveInputCommand;
    }
}
1 reference
private void ExecuteSaveInputCommand(object obj)

```

Δήλωση κλάσεων των δεδομένων του Viewmodel

Δήλωση και κατασκευή εντολών. Βλέπε εντολή SaveInput Command. Η εντολή εκτελείται στο ExecuteSaveInput Command. Η εντολή ενεργοποιείται όταν πατηθεί το κουμπί που έχει γίνει το Bind στο View, όπως βλέπετε στο κάτω κουτί

```
<Button x:Name="SaveData" Width="100" HorizontalAlignment="Left"
        Command="{Binding SaveInputCommand}">
    <TextBlock Text="Save"/>
</Button>
```

Εικόνα 7.17: Δήλωση κλάσεων και μεταβλητών , View Model

7.3) Vacation Planning

Στο κεφάλαιο 7.4.1, 7.4.2 γίνεται εκτενής ανάλυση του model και της δομής του View Model το οθόνης Vacation Planning.

Στο κεφάλαιο 7.4.3 παρουσιάζεται η εντολή δυναμικής εισαγωγής δεδομένων στις παραμέτρους του προβλήματος ανάθεσης αδειών που αναλύθηκε στο κεφάλαιο 3.

Στο κεφάλαιο 7.4.4 παρουσιάζεται η δομή της εντολής CalculateVacationPlanning_Cplex και δίνεται έμφαση στην εξαγωγή της λύσης της βελτιστοποίησης και η εισαγωγή της σε κλάσεις του Model . Σκοπός μας είναι να εμφανίσουμε τα αποτελέσματα στα Data Grids της 2ης καρτέλας (Optimization Results) της οθόνης Vacation Planning. Βλέπε (6.4.2) στο manual για καλύτερη κατανόηση της δομής και τον λειτουργιών της οθόνης.

7.3.1) Vacation Planning, Ανάλυση Model

Παρουσιάζονται όλες οι κλάσεις της οθόνης με επεξηγηματικά σχόλια εντός του κώδικα .

```
public class VacationPlanningInputData : RecordBaseModel
{
    #region General

    //Id Πειράματος
    public int VPId { get; set; }

    //Κωδικός Πειράματος
    public string VPCode { get; set; }

    //Περιγραφή Πειράματος
    public string VPDescr { get; set; }

    //Επάγγελμα Υπαλλήλου
    public BasicEnums.EmployeeType EmployeeType { get; set; }
    //Λογική Ανάθεσης αδειών
    public BasicEnums.VPLogicType VPLogicType { get; set; }

    //Bmax στη μορφοποίηση : Μέγιστος αριθμός ικανοποιημένων προτάσεων άδειας για κάθε
    υπάλληλο
    public int MaxSatisfiedBids { get; set; }

    //Se στη μορφοποίηση: Τιμή διαχωρισμού μεταξύ 2 ικανοποιημένων προτάσεων
    public int SeparValue { get; set; }
    //Επιλεγμένο Χρονοδιάγραμμα
    public ReqScheduleInfoData Schedule { get; set; }
    //Όλοι οι υπάλληλοι του πειράματος με επάγγελμα = επιλεγμένο Employee Type
    public ObservableCollection<EmployeeData> Employees { get; set; }

    //Βοηθητική λίστα με τις ημερομηνίες του προβλήματος
    public DateTime[] Dates { get; set; }

    //Βοηθητική λίστα με τις ημερομηνίες του προβλήματος σε μορφή string
    public string[] DatesStr { get; set; }
    #endregion

    #region Παράμετροι Βελτιστοποίησης

    //LLt στη μορφοποίηση : Αριθμός υπαλλήλων που επιτρέπεται να βρίσκονται σε άδεια
    την ημέρα t, t∈T
    public Dictionary<int, int> LL_Dict { get; set; }

    //Ei στη μορφοποίηση: Ημέρες Άδειας που δικαιούται ο υπάλληλος i, i∈I
```

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```
public Dictionary<int, int> MaxD_Dict { get; set; }

//JBi στη μορφοποίηση : Αριθμός Προτάσεων άδειας υπαλλήλου
public Dictionary<int, int> N_Dict { get; set; }

//NDBij στη μορφοποίηση : Ημέρες άδειας της πρότασης άδειας  $j$  του υπαλλήλου  $i$   $i \in I$ ,  $j \in J$ 
//Αν ο τύπος άδειας είναι Min-Max τότε οι ημέρες άδειας = Μέγιστες ημέρες άδειας
'
public Dictionary<(int, int), int> NDays_Dict { get; set; }

//DFij στη μορφοποίηση : Ημερομηνία έναρξης της πρότασης άδειας  $j$  του υπαλλήλου  $i$ ,  $i \in I$ ,  $j \in J$ 
public Dictionary<(int, int), int> DateFrom_Dict { get; set; }

//DTij στη μορφοποίηση : Ημερομηνία λήξης της πρότασης άδειας  $j$  του υπαλλήλου  $i$ ,  $i \in I$ ,  $j \in J$ 
public Dictionary<(int, int), int> DateTo_Dict { get; set; }

//RBij στη μορφοποίηση : Αριθμός Non-Specific αδειών της πρότασης άδειας  $j$  του υπαλλήλου  $i$ ,  $i \in I$ ,  $j \in J$ 
public Dictionary<(int, int), int> RBids_Dict { get; set; }

//ZBijrz Αριθμός Specific αδειών της non-specific άδειας  $r$  της πρότασης άδειας  $j$  του υπαλλήλου  $i$ ,  $i \in I$ ,  $j \in J$ 
public Dictionary<(int, int, int), int> ZBids_Dict { get; set; }

#endregion
}
```

VacationPlanningOutputData

```
public class VacationPlanningOutputData : RecordBaseModel
{
    #region Optimization Data Output
    //Τιμή Αντικειμενικής συνάρτησης
    public double ObjValue { get; set; }

    //Καρτέλα Αποτελεσμάτων : Πλάνο  $Y_{ij}$ 
    public ObservableCollection<VPYijResultsData> VPYijResultsDataGrid { get; set; }

    //Καρτέλα Αποτελεσμάτων : Πλάνο  $Y_{ijrz}$ 
    public ObservableCollection<VPYijrzResultsData> VPYijrzResultsDataGrid { get; set; }
}

//Καρτέλα Αποτελεσμάτων : Πλάνο  $X_{it}$ 
public ObservableCollection<VPXitResultsData> VPXitResultsDataGrid { get; set; }

//Καρτέλα Αποτελεσμάτων : RLLt στην μορφοποίηση
public ObservableCollection<VPXtResultData> VPRLLtResultsDataGrid { get; set; }

//Καρτέλα Αποτελεσμάτων : Entitlement στην μορφοποίηση
public ObservableCollection<EmployeeData> EmpLeaveStatusData { get; set; }

#endregion
}

public class VPYijResultsData
{
}
```

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```
// Υπάλληλος i
public EmployeeData Employee { get; set; }

// Προτίμηση άδειας j
public LeaveBidsDataStatic LeaveBidData { get; set; }

//Όνομα προτίμησης άδειας Vij
public string Vij { get; set; }

//Όνομα μεταβλητής απόφασης Rijr
public string Rijr { get; set; }

//Όνομα μεταβλητής Vijrz
public string Vijrz { get; set; }

//Ακριβής ημερομηνία παραγόμενης άδειας
public string Date { get; set; }

//Τιμή μεταβλητής απόφασης , χορήγηση άδειας
public double VijrzFlag { get; set; }

}
```

```
public class VPXitResultsData
{
    //Κωδικός Υπαλλήλου
    public EmployeeData Employee { get; set; }

    //Όνομα μεταβλητης αποφασης Xit
    public string Xit { get; set; }

    //Ημερομηνία
    public string Date { get; set; }

    //Τιμή μεταβλήτης απόφασης Xit
    public bool Granted{ get; set; }
}
```

```
public class VPRRLtResultData
{
    //Ημερομηνία
    public string Date { get; set; }

    //Limit Line πριν την βελτιστοποίηση
    public string LLi { get; set; }

    //Υπολειπομενο Limit Line μετά την βελτιστοποίηση
    public int Remaining LimitLine { get; set; }
}
```

```
public class VPCGInputData : RecordBaseModel
{
    public string[] Dates { get; set; }

    //Υπολειπόμενο LimitLine ανά ημέρα σε Model class
    public ObservableCollection<VPRRLtResultData>
    VP_Remaining_LimitLine_ResultsDataGrid { get; set; }

    #region Dictionaries

    //REi στη μορφοποίηση : Υπολειπόμενες ημέρες άδειας δυο δικαιούται ο υπάλληλος i,
    i∈I
}
```

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```
//να βρίσκονται σε άδεια την ημέρα t,  $t \in T$ 
public Dictionary<int, int> LeaveDays { get; set; }

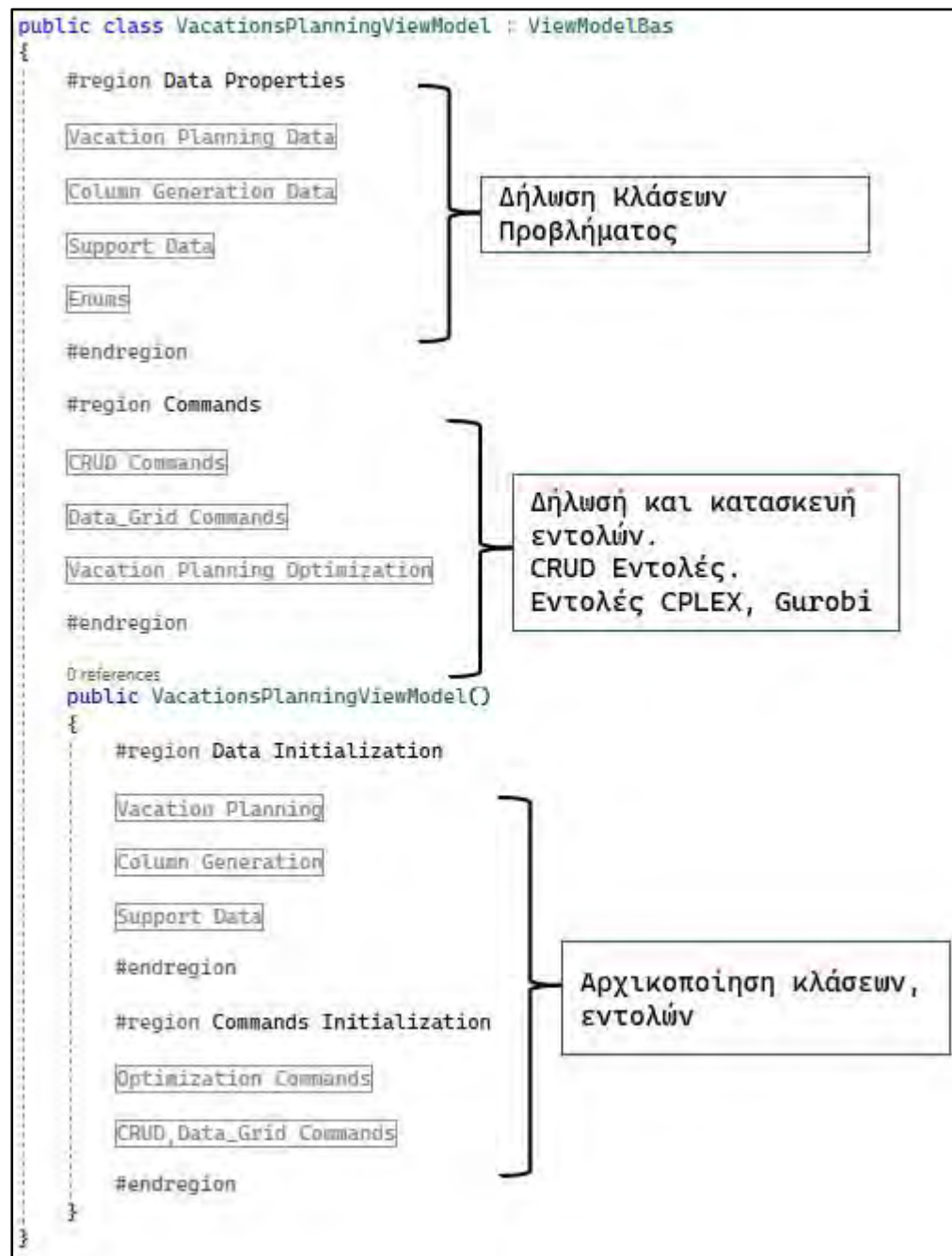
//RRLlt στη μορφοποίηση : Υπολειπόμενος Αριθμός υπαλλήλων που επιτρέπεται να
βρίσκονται σε άδεια την ημέρα t,  $t \in T$ 
public Dictionary<int, int> LLiDict { get; set; }

#endregion
}
```

```
public class VPCGOutputData : RecordBaseModel
{
    public double ObjValue;    } //Τιμή της αντικειμενικής συνάρτησης της
μορφοποίησης
```

7.3.2) Vacation Planning, Ανάλυση View Model

Στην εικόνα (7.18) παρουσιάζεται η οργάνωση της δομής του view model .



Εικόνα 7.18: Vacation Planning View model, δομή

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στην εικόνα (7. 19) παρουσιάζεται η δομή της εντολής ExecuteCalculateVacationPlanning_Cplex του View model .Η εντολή αποσκοπεί στην δυναμική εισαγωγή παραμέτρων στο πρόβλημα ανάθεσης αδειών και επίλυσης του (Κεφάλαιο 3).

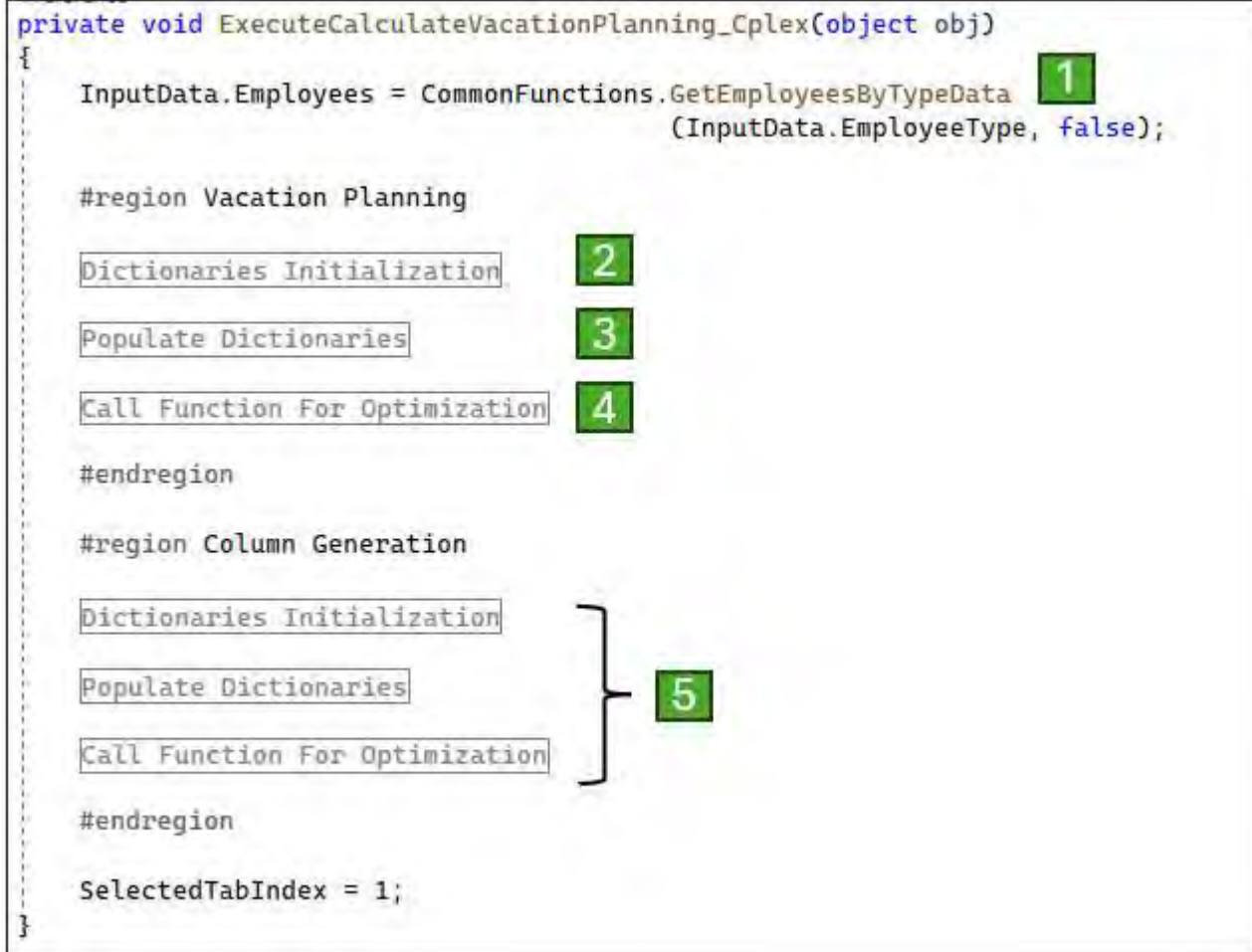
Επίσης τα δεδομένα εξαγωγής της του προβλήματος ανάθεσης αδειών χρησιμοποιούνται ως δεδομένα εισαγωγής στο Column Generation (Κεφάλαιο 4)

```
private void ExecuteCalculateVacationPlanning_Cplex(object obj)
{
    InputData.Employees = CommonFunctions.GetEmployeesByTypeData
                                   (InputData.EmployeeType, false);

    #region Vacation Planning
    Dictionaries Initialization
    Populate Dictionaries
    Call Function For Optimization
    #endregion

    #region Column Generation
    Dictionaries Initialization
    Populate Dictionaries
    Call Function For Optimization
    #endregion

    SelectedTabIndex = 1;
}
```



Εικόνα 7.19: Calculate Vacation planning 1

1) GetEmployeesByTypeData :

Καλεί από τη βάση όλους τους υπαλλήλους με το επιλεγμένο επάγγελμα.

2) Dictionaries Initialization :

Αρχικοποίηση των παραμέτρων της βελτιστοποίησης ανάθεσης αδειών. Παρατίθεται εικόνα (7.19)

```
#region Dictionaries Initialization

InputData.LLi_Dict = new Dictionary<int, int>();
InputData.MaxD_Dict = new Dictionary<int, int>();
InputData.N_Dict = new Dictionary<int, int>();
InputData.NDays_Dict = new Dictionary<int, int>, int>();
InputData.DateFrom_Dict = new Dictionary<int, int>, int>();
InputData.DateTo_Dict = new Dictionary<int, int>, int>();

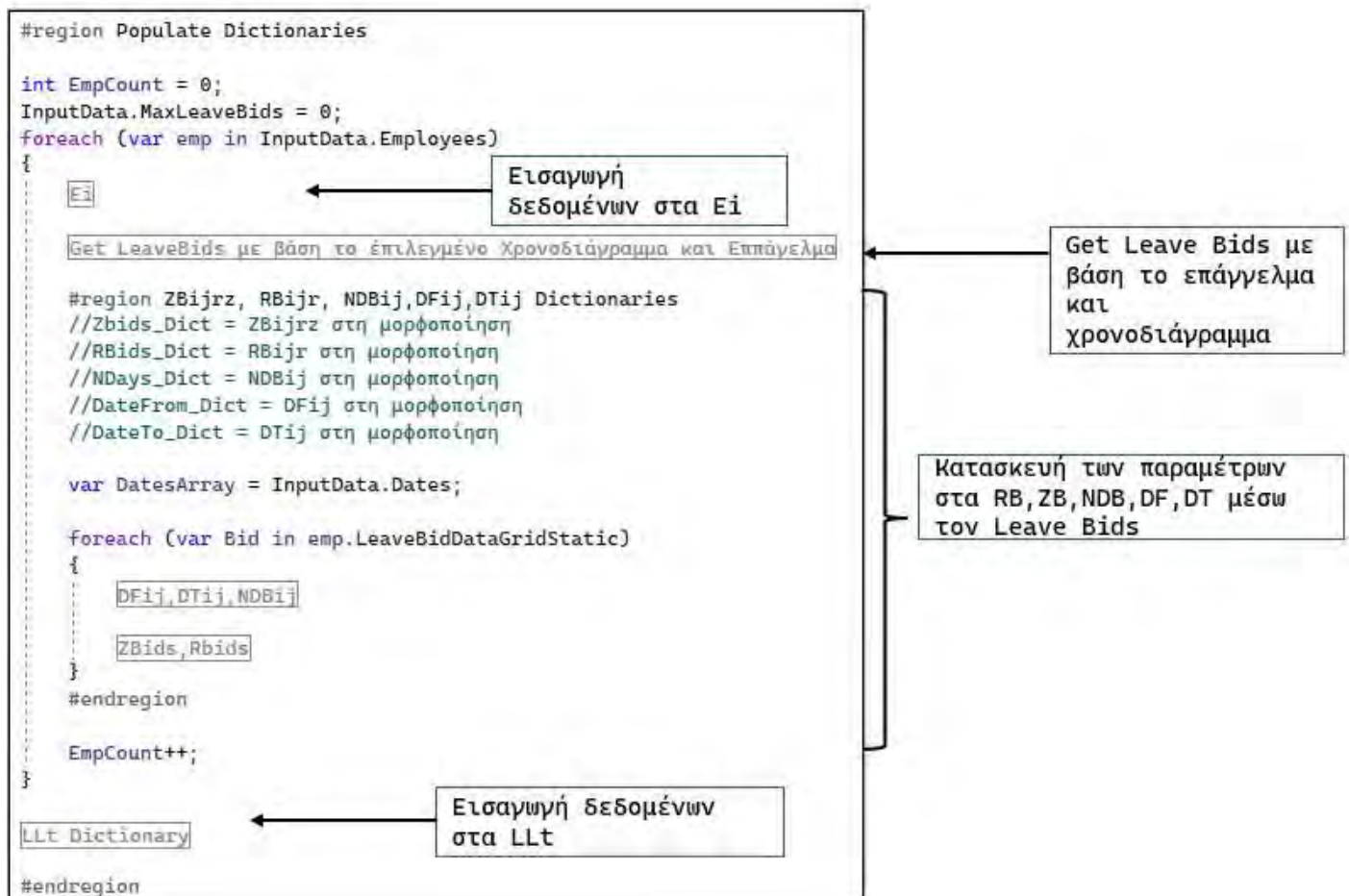
InputData.RBids_Dict = new Dictionary<int, int>, int>();
InputData.ZBids_Dict = new Dictionary<int, int, int>, int>();
```

Εικόνα 7.19: Calculate Vacation planning 2

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3) Populate Dictionaries :

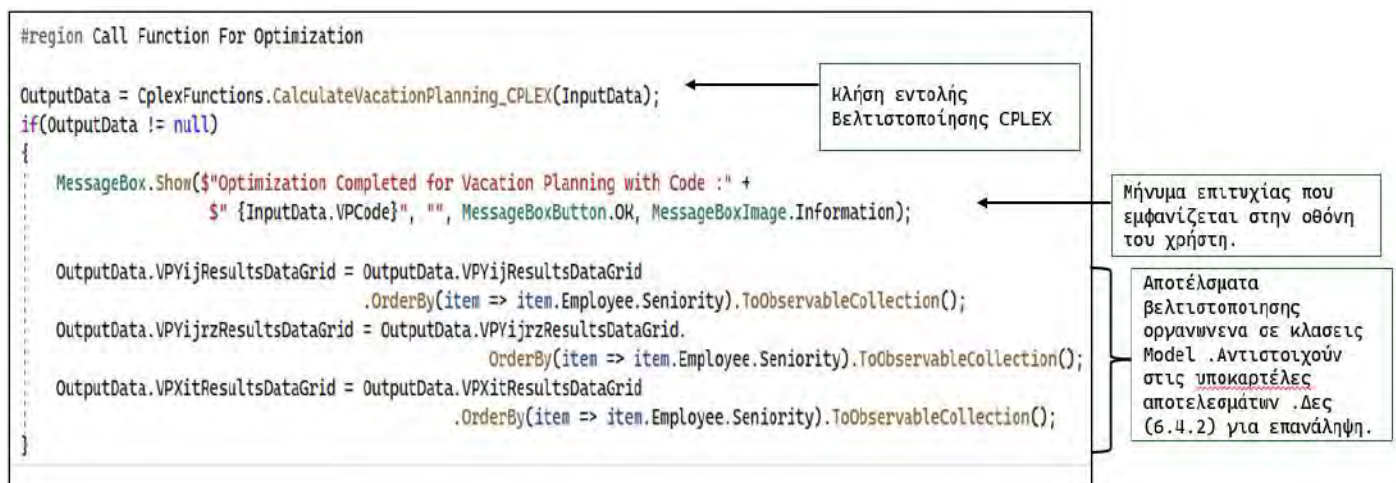
Εισαγωγή των δεδομένων από τις classes του view model στις παραμέτρους / Dictionaries και δείκτες του προβλήματος. Παρατίθεται εικόνα δομής του region. Εκτενής ανάλυση του region παρουσιάζεται στο υποκεφάλαιο 7.4.3.



Εικόνα 7.19: Calculate Vacation planning 3

4) Call function for optimization :

Παρουσίαση της εντολής

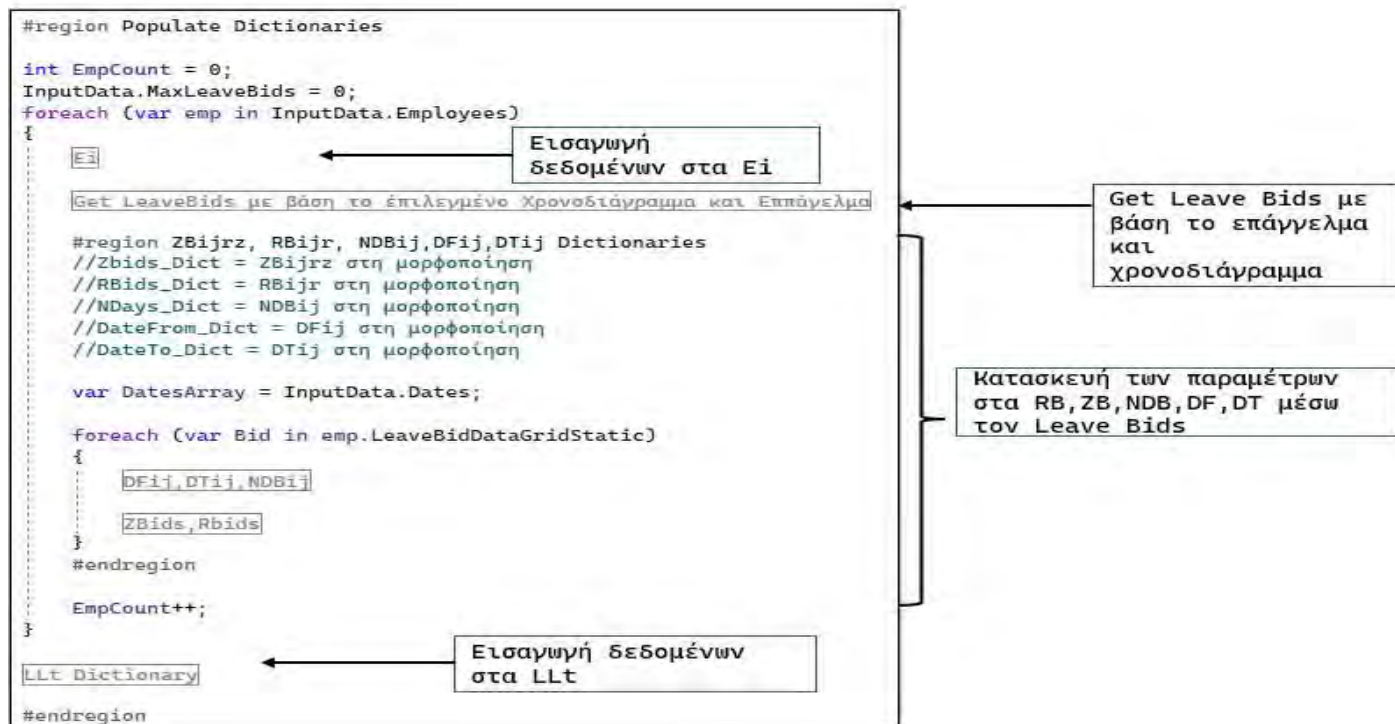


Εικόνα 7.20 Calculate Vacation planning 4

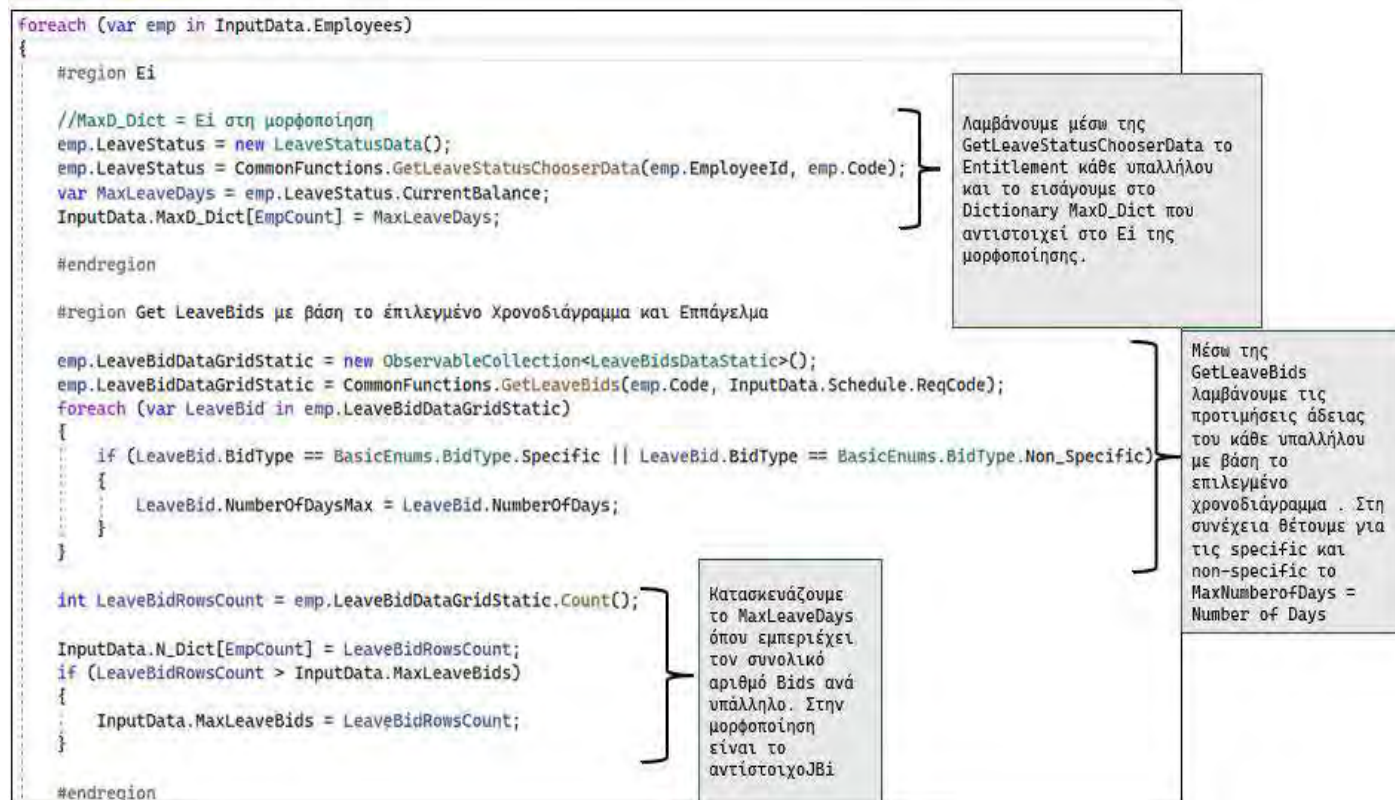
7.3.3) Vacation Planning Optimization, Δυναμική εισαγωγή παραμέτρων.

Στο κεφάλαιο 7.3.3 παρουσιάζεται εκτενής ανάλυση το region Populate Dictionaries που παρουσιάστηκε εισαγωγικά στο κεφάλαιο 7.3.2. Σκοπός του region είναι η κατασκευή των παραμέτρων του προβλήματος ανάθεσης αδειών από τις Model classes του Vacation Planning view model.

Παρατίθενται εικόνες με ανάλυση του region .



Εικόνα 7.21 VP εισαγωγή δεδομένων 1



Εικόνα 7.22 VP εισαγωγή δεδομένων 2

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```

#region ZBijrz, RBijr, NDBij, DFij, DTij Dictionaries
//Zbids_Dict = ZBijrz στη μορφοποίηση
//RBids_Dict = RBijr στη μορφοποίηση
//NDbids_Dict = NDBij στη μορφοποίηση
//DateFrom_Dict = DFij στη μορφοποίηση
//DateTo_Dict = DTij στη μορφοποίηση

Var DatesArray = InputData.Dates;

foreach (var Bid in emp.LeaveBidDataGridStatic)
{
    #region DFij,DTij,NDBij

    int DateFrom_Index = Array.IndexOf(DatesArray, Bid.DateFrom);
    int DateTo_Index = Array.IndexOf(DatesArray, Bid.DateTo);

    InputData.DateFrom_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = DateFrom_Index;
    InputData.DateTo_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = DateTo_Index;

    InputData.NDbids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = Bid.NumberOfDaysMax;

    #endregion

    ZBids, RBids
        
```

Εισάγουμε τιμές στα Dictionaries DateFrom_Dict, DateTo_Dict Χρησιμοποιώντας την λίστα DatesArray που περιέχει τις ημερομηνίες του χρονικού ορίζοντα σε μορφή int και όχι datetime. Ταυτόχρονα Γεμίζουμε το dictionary NDbids_Dict.

Εικόνα 7.23 VP εισαγωγή δεδομένων 3

```

#region ZBids, RBids

if (Bid.BidType == BasicEnums.BidType.Specific)
{
    InputData.RBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = 1;
    InputData.ZBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1, 0)] = 1;
}
else if (Bid.BidType == BasicEnums.BidType.Non_Specific)
{
    // Calculate the number of combinations
    TimeSpan range = Bid.DateTo - Bid.DateFrom;
    int totalDaysInRange = (int)range.TotalDays;

    int Z = totalDaysInRange - Bid.NumberOfDays + 2;

    InputData.RBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = 1;
    InputData.ZBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1, 0)] = Z;
}
else if (Bid.BidType == BasicEnums.BidType.Min_Max)
{
    TimeSpan range = Bid.DateTo - Bid.DateFrom;
    int totalDaysInRange = (int)range.TotalDays;

    var Min = Bid.NumberOfDaysMin;
    var Max = Bid.NumberOfDaysMax;

    InputData.RBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1)] = Max - Min + 1;

    for (int i = 0; i < Max - Min + 1; i++)
    {
        Bid.NumberOfDays = Max - i;
        int Z = totalDaysInRange - Bid.NumberOfDays + 2;

        InputData.ZBids_Dict[(emp.Seniority - 1, Bid.PriorityLevel - 1, i)] = Z;
    }
}
        
```

Specific
Θέτουμε το RBijr και το ZBijrz = 1

NON-Specific :
Θέτουμε το RBijr =1 αφού μια non-specific παράγει 1 non-specific

Zbijrz :
Υπολογίζουμε το $Z = DT-DF -NDB +2$.

Min max :
RBijr :
 $R = Max - min + 2$.

Zbijrz :
 $Z = DT-DF -NDB +2$ για κάθε R.

Εικόνα 7.24 VP εισαγωγή δεδομένων 4

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

7.3.4) Vacation Planning Optimization, Εξαγωγή δεδομένων βέλτιστης λύσης.

Στο κεφάλαιο αυτό αρχικά θα παρουσιαστεί μια εικόνα περιγραφής της δομής την CalculateVacationPlanning_Cplex.

Στη συνέχεια θα παρουσιαστούν πληθώρα φωτογραφιών για την ανάλυση την εξαγωγής των αποτελεσμάτων της βελτιστοποίησης και εισαγωγής τους στις μεταβλητές της VacationPlanningOutputData.

```
public VacationPlanningOutputData CalculateVacationPlanning_CPLEX(VacationPlanningInputData InputData)
{
    VacationPlanningOutputData OutputData = new VacationPlanningOutputData();

    try
    {
        #region Optimization Algorithm

        Indexes
        Parameters
        Cplex Model Initialization
        Decision Variables
        Objective Function
        Constrains
        Check Bids (Strict Seniority, Fair Priority, Bid by Bid)

        #endregion

        Support Arrays
        Output Initialization
        Retrieve Optimization Results

        return OutputData;
    }
    catch (ILOG.Concert.Exception ex)
    {
        Console.WriteLine("A CPLEX error occurred: " + ex.Message);
        return OutputData;
    }
    catch (System.Exception ex)
    {
        Console.WriteLine("A System error occurred: " + ex.Message);
        return OutputData;
    }
}
```

Μοντελοποίηση Παρουσιάστηκε στην ενότητα 3.4 .

- Βοηθητικά Arrays
- Αρχικοποίηση των μεταβλητών της OutputData
- Εισαγωγή της λύσης σε βοηθητικά Dictionaries και μεταφορά τους στην Model Class VacationPlanningOutputData

Εικόνα 7.25 VP εξαγωγή δεδομένων 1

```
#region Support Arrays

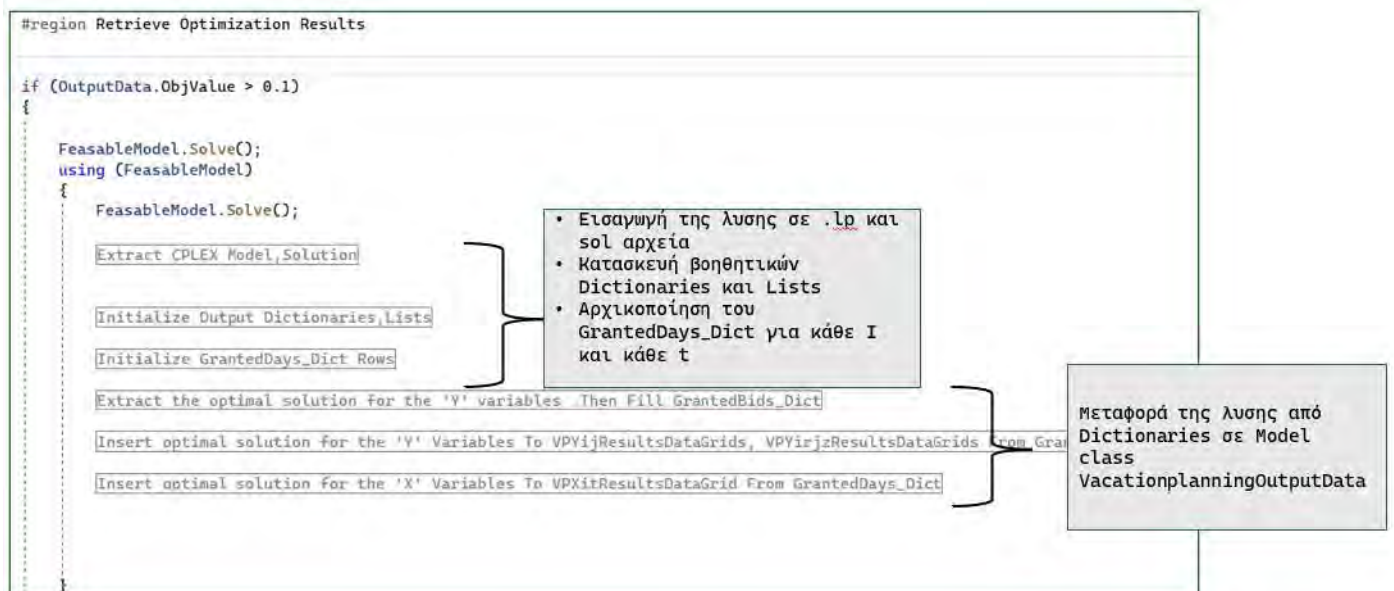
string[] Employees_Array = InputData.Employees.Select(d => d.Code).ToArray();
string[] Dates_Array = InputData.DatesStr;

#endregion

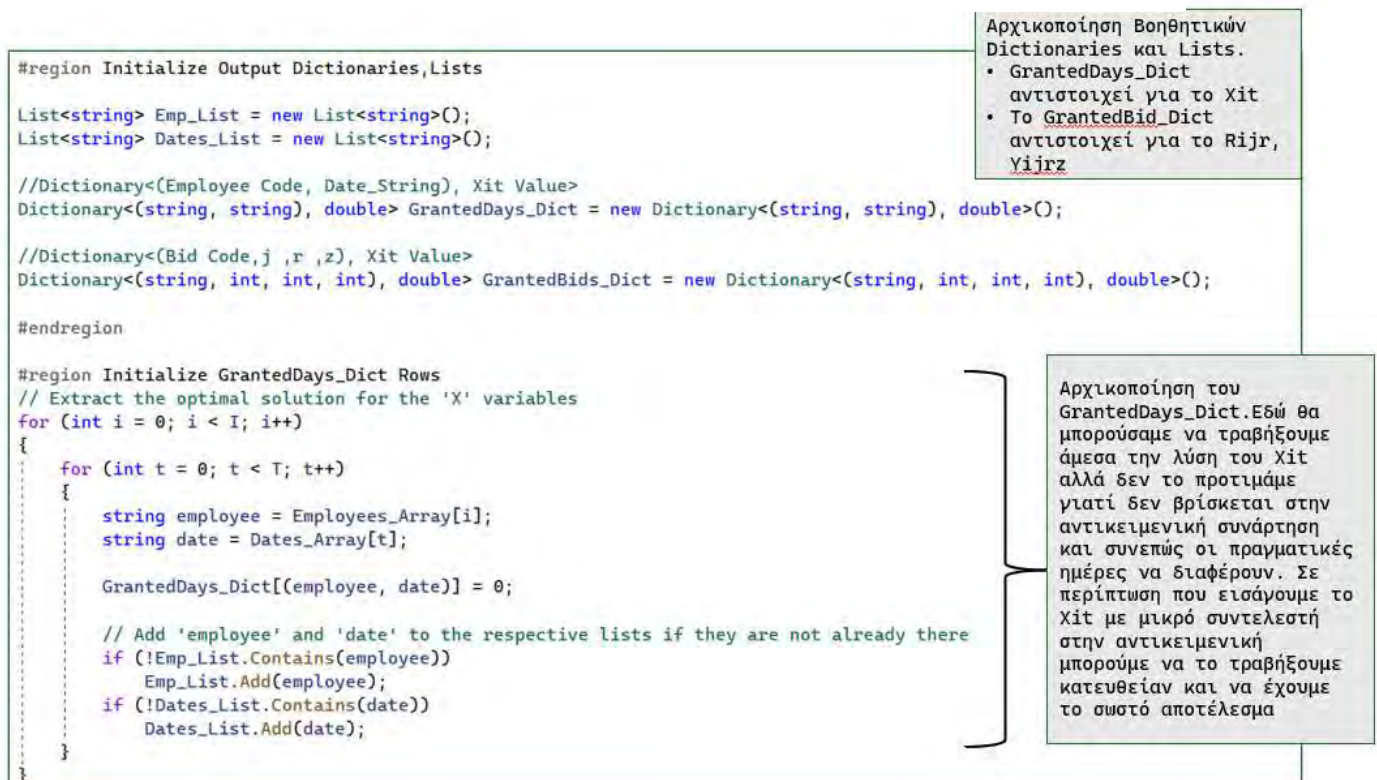
#region Output Initialization
OutputData.VPYijResultsDataGrid = new ObservableCollection<VPYijResultsData>();
OutputData.VPYijrzResultsDataGrid = new ObservableCollection<VPYijResultsData>();
OutputData.VPXitResultsDataGrid = new ObservableCollection<VPXiResultsData>();
OutputData.VPLLiResultsDataGrid = new ObservableCollection<VPXiResultsData>();
OutputData.EmpLeaveStatusData = new ObservableCollection<EmployeeData>();
#endregion
```

Εικόνα 7.26 VP εξαγωγή δεδομένων 2

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ



Εικόνα 7.27 VP εξαγωγή δεδομένων 3



Εικόνα 7.28 VP εξαγωγή δεδομένων 3

Ο κώδικας για τα regions :

Extract the optimal solution for the 'Y' variables .Then Fill GrantedBids_Dict

Insert optimal solution for the 'Y' Variables To VPYijResultsDataGrids, VPYirjzResultsDataGrids From GrantedBids_Dict

Insert optimal solution for the 'X' Variables To VPXitResultsDataGrid From GrantedDays_Dict

Εισάγεται σε μορφή κώδικα και όχι σε εικόνες .

```
#region Extract the optimal solution for the 'Y' variables .Then Fill GrantedBids_Dict
for (int i = 0; i < I; i++)
{
    for (int j = 0; j < JB[i]; j++)
    {
        var specificEmployee = InputData.Employees.FirstOrDefault(emp => emp.Code
        == Employees_Array[i]);
        var Rvalue = RB[(i, j)];

        for (int r = 0; r < Rvalue; r++)
        {
            var Zvalue = ZB[(i, j, r)];

            double rValue = FeasableModel.GetValue(R[i,j,r]);

            for (int z = 0; z < Zvalue; z++)
            {
                string employee = Employees_Array[i];
                int bidIndex = j;

                double yValue = FeasableModel.GetValue(Y[i, j, r,z]);

                // Store the optimal 'Y' value in the data structure
                GrantedBids_Dict[(employee, bidIndex, r, z)] = yValue;
            }
        }
    }
}

#endregion

#region Insert optimal solution for the 'Y' Variables To VPYijResultsDataGrids,
VPYirjzResultsDataGrids From GrantedBids_Dict
Console.WriteLine("\nOptimal Solution for Y Variables:");
for (int i = 0; i < I; i++)
{
    //Retrieve Employee
    var EmployeeCode = Emp_List[i];
    var Current_Employee = InputData.Employees.FirstOrDefault(emp => emp.Code ==
EmployeeCode);

    int TotalGrantedDays = 0;

    for (int j = 0; j < JB[i]; j++)
    {
        var Current_Bid = Current_Employee.LeaveBidDataGridStatic[j];
        for (int r = 0; r < RB[(i, j)]; r++)
        {
            for (int z = 0; z < ZB[(i, j, r)]; z++) //allagh
            {

                #region Populate VPYijResultsDataGrid

                //Retrieve YValue from Dictionary
                double yValue = GrantedBids_Dict[(Current_Employee.Code, j, r,
z)];

                string Yij_String = $"Y{(i + 1)}{(j + 1)}";

                var existingYijRecord =
OutputData.VPYijResultsDataGrid.FirstOrDefault(record => record.Yij == Yij_String);
```

```

        if (existingYijRecord != null)
        {
            #region Update
            if (existingYijRecord.YijFlag == 0 && yValue == 1)
            {
                existingYijRecord.YijFlag = yValue;

                OutputData.VPYijResultsDataGrid.Remove(existingYijRecord);
                OutputData.VPYijResultsDataGrid.Add(existingYijRecord);
            }
        }
        else
        {
            #region Initialize Yij Record

            VPYijResultsData yijDataRecord = new VPYijResultsData();
            yijDataRecord.LeaveBidData = new LeaveBidsDataStatic();
            #endregion

            #region Refresh NumberOfDays ,NumberOfDaysMin
            ,NumberOfDaysMax Based on BidType

            var NumberOfDays = Current_Bid.NumberOfDays;
            var NumberOfDaysMax = Current_Bid.NumberOfDaysMax;
            var NumberOfDaysMin = Current_Bid.NumberOfDaysMin;

            var BidType = Current_Bid.BidType;
            if (BidType == BasicEnums.BidType.Min_Max)
            {
                NumberOfDays = 0;
            }
            else if (BidType == BasicEnums.BidType.Non_Specific ||
            BidType == BasicEnums.BidType.Specific)
            {
                NumberOfDaysMax = 0;
                NumberOfDaysMin = 0;
            }

            #endregion

            #region Insert Values To VPYijResultsDataGrid

            yijDataRecord.Yij = $"Y{(i + 1)}{(j + 1)}";
            yijDataRecord.Rijr = $"R{(i + 1)}{(j + 1)}{(r + 1)}";
            yijDataRecord.Yijrz = $"Y{(i + 1)}{(j + 1)}{(r + 1)}{(z
            + 1)}";

            yijDataRecord.YijFlag = yValue;
            yijDataRecord.Employee = Current_Employee;
            yijDataRecord.LeaveBidData = Current_Bid;
            yijDataRecord.DateFrom = Current_Bid.DateFrom;
            yijDataRecord.DateTo = Current_Bid.DateTo;
            yijDataRecord.DateFromStr =
            Current_Bid.DateFrom.ToString("dd/MM/yyyy");
            yijDataRecord.DateToStr =
            Current_Bid.DateTo.ToString("dd/MM/yyyy");

            yijDataRecord.NumberOfDays = NumberOfDays;
            yijDataRecord.NumberOfDaysMax = NumberOfDaysMax;
            yijDataRecord.NumberOfDaysMin = NumberOfDaysMin;

            //Add record to Datagrid

```

```

        OutputData.VPYijResultsDataGrid.Add(yijDataRecord);

        #endregion

    }

    #endregion

    #endregion

    #region Populate VPYijrzResultsDataGrid

    #region Initialize Yirz Record

    VPYijResultsData yijrzDataRecord = new VPYijResultsData();
    yijrzDataRecord.LeaveBidData = new LeaveBidsDataStatic();
    #endregion

    #region Edit Dates, NumberOfDays based on r,z
    DateTime DateFrom = Current_Bid.DateFrom.AddDays(z);
    var Yirjz_NumberOfDays = Current_Bid.NumberOfDaysMax - r;
    DateTime DateTo = DateFrom.AddDays(Yirjz_NumberOfDays - 1);
    #endregion

    #region Insert Values To VPYijrzResultsDataGrid

    yijrzDataRecord.Yij = $"Y{(i + 1)}{(j + 1)}";
    yijrzDataRecord.Rijr = $"R{(i + 1)}{(j + 1)}{(r + 1)}";
    yijrzDataRecord.Yijrz = $"Y{(i + 1)}{(j + 1)}{(r + 1)}{(z +
1)}}";

    yijrzDataRecord.YijFlag = yValue;
    yijrzDataRecord.Employee = Current_Employee;
    yijrzDataRecord.LeaveBidData = Current_Bid;
    yijrzDataRecord.DateFrom = DateFrom;
    yijrzDataRecord.DateTo = DateTo;
    yijrzDataRecord.NumberOfDays = Yirjz_NumberOfDays;
    yijrzDataRecord.DateFromStr = DateFrom.ToString("dd/MM/yyyy");
    yijrzDataRecord.DateToStr = DateTo.ToString("dd/MM/yyyy");

    //Add record to Datagrid
    OutputData.VPYijrzResultsDataGrid.Add(yijrzDataRecord);
    #endregion

    #region Update TotalGrantedDays
    if (yValue == 1)
    {
        TotalGrantedDays = TotalGrantedDays +
Yirjz_NumberOfDays;
    }
    #endregion

    #endregion

    #region Extract the optimal solution for the 'X' variables
    .Then Fill GrantedDays_Dict
    if (yValue == 1)
    {
        for (DateTime date = DateFrom; date <= DateTo; date =
date.AddDays(1))
        {
            var SelectedDate_Str =

date.ToString("dd/MM/yyyy");

            GrantedDays_Dict[(Current_Employee.Code,

```

7. ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

```
SelectedDate_Str)] = 1;

        }
    }
#endregion

}

}

var UpdatedEmp = InputData.Employees.FirstOrDefault(emp => emp.Code ==
Current_Employee.Code);
UpdatedEmp.LeaveStatus.ProjectedBalance = UpdatedEmp.LeaveStatus.CurrentBalance -
TotalGrantedDays;
OutputData.EmpLeaveStatusData.Add(UpdatedEmp);

}
#endregion

#region Insert optimal solution for the 'X' Variables To VPXitResultsDataGrid From
GrantedDays_Dict
foreach (var employee in Emp_List)
{
    foreach (var date in Dates_List)
    {
        double xValue = GrantedDays_Dict.ContainsKey((employee, date)) ?
GrantedDays_Dict[(employee, date)] : 0.0;

        #region Populate VP Xij
        VPXijResultsData singleDataRecord = new VPXijResultsData();

        singleDataRecord.Xit = $"X{(Array.IndexOf(Employees_Array, employee) +
1)}{(Array.IndexOf(Dates_Array, date) + 1)}";
        singleDataRecord.XitFlag = xValue;
        singleDataRecord.Date = date;

        var SpecificEmployee = InputData.Employees.FirstOrDefault(emp => emp.Code
== employee);
        singleDataRecord.Employee = SpecificEmployee;

        OutputData.VPXitResultsDataGrid.Add(singleDataRecord);
        #endregion
    }
}

}
#endregion
```

8.1) Συμπεράσματα

Η διπλωματική αυτή εργασία παρουσιάζει τις μεθοδολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη ενός συστήματος διαχείρισης αδειών και κατανομής πληρώματος. Η εφαρμογή αποδείχθηκε αποτελεσματική τόσο στην εισαγωγή δεδομένων όσο και στην εμφάνιση των αποτελεσμάτων μέσω ενός φιλικού προς το χρήστη γραφικού περιβάλλοντος. Η χρήση της βάσης δεδομένων SQL Server διασφάλισε την αξιόπιστη αποθήκευση και ανάκτηση των δεδομένων, ενώ οι optimization solvers Cplex και Gurobi επέτρεψαν την επίλυση σύνθετων προβλημάτων βελτιστοποίησης με ακρίβεια και ταχύτητα.

Κατά τη διάρκεια της ανάπτυξης της εφαρμογής, η υιοθέτηση της αρχιτεκτονικής Model-View-View Model (MVVM) διευκόλυνε τη διαχείριση του κώδικα και την ενσωμάτωση νέων λειτουργιών. Τα αποτελέσματα των δοκιμών έδειξαν ότι η εφαρμογή μπορεί να προσαρμοστεί σε διαφορετικές ανάγκες και περιβάλλοντα, καθιστώντας την ένα πολύτιμο εργαλείο. Συνολικά, η εργασία αποδεικνύει ότι ένα σύστημα όπως το Vacation Management System (VMS) μπορεί να βελτιώσει σημαντικά την αποδοτικότητα και την οργάνωση των διαδικασιών, προσφέροντας παράλληλα μια αποτελεσματική και ευέλικτη λύση για τον κλάδο των αερομεταφορών.

8.2) Βιβλιογραφία

- 1) Associate Prof. George Kozanidis: An integrated column generation solution framework for optimal aircrew vacation planning subject to seniority ranking and priority preference satisfaction.
- 2) Associate Prof. George Kozanidis: Set-cover master problem formulations for maximum duty coverage in branch & price solution algorithms for optimal aircrew scheduling.