ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΜΜΕ

Πτυχιακή εργασία

«ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΦΑΡΜΟΓΗΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΑΤΖΕΝΤΑΣ ΜΕ ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΔΙΕΠΑΦΗΣ ΚΑΝΟΝΤΑΣ ΧΡΗΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ VISUAL STUDIO ΤΗΣ MICROSOFT ΚΑΙ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ VISUAL C#.»

Επιβλέπων καθηγητής:

Βησσαρίων Φερεντίνος

Φοιτήτρια:

Φωτεινή Σταματάκη

ΠΥΡΓΟΣ 2018

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η πτυχιακή εργασία με θέμα:

«ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΦΑΡΜΟΓΗΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΑΤΖΕΝΤΑΣ ΜΕ ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΔΙΕΠΑΦΗΣ ΚΑΝΟΝΤΑΣ ΧΡΗΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ VISUAL STUDIO ΤΗΣ MICROSOFT ΚΑΙ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ VISUAL C#.»

της φοιτήτριας του Τμήματος ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΜΜΕ

Φωτεινή Σταματάκη

παρουσιάστηκε δημόσια και εξετάσθηκε στο Τμήμα ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΜΜΕ στις

18 / 09 / 2018

Ο ΕΠΙΒΛΕΠΩΝ

Βησσαρίων Φερεντίνος

Ο ΠΡΟΕΔΡΟΣ ΤΟΥ ΤΜΗΜΑΤΟΣ

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΜΗ ΛΟΓΟΚΛΟΠΗΣ

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Ακόμα δηλώνω ότι αυτή η γραπτή εργασία προετοιμάστηκε από εμένα προσωπικά και αποκλειστικά και ειδικά για την συγκεκριμένη πτυχιακή εργασία και ότι θα αναλάβω πλήρως τις συνέπειες εάν η εργασία αυτή αποδειχθεί ότι δεν μου ανήκει.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΣΠΟΥΔΑΣΤΗ 1	ΑΜ	ΥΠΟΓΡΑΦΗ
Φωτεινή Σταματάκη	1124	

Σύνοψη

Η ραγδαία άνοδος των σύγχρονων ρυθμών διαβίωσης καθιστά τη χρήση μιας ηλεκτρονικής agenda απαραίτητη.

- Η αποθήκευση των καθημερινών λειτουργιών και υποχρεώσεων σε ηλεκτρονική μορφή,
- Η διαχείριση του χρόνου,
- Η συνεχής ενημέρωση για γεγονότα όπως γενέθλια, ραντεβού, εορτές

είναι μερικά από τα ζητήματα που δύναται να λύσει η τεχνολογία του ηλεκτρονικού organizer.

Η παρούσα πτυχιακή εργασία πραγματεύεται την δημιουργία και την παρουσίαση ενός ηλεκτρονικού organizer. Η εφαρμογή που περιγράφεται παρακάτω είναι υλοποιημένη με τη γλώσσα προγραμματισμού **C#** στην πλατφόρμα **Visual Studio.**

Abstract

The rapid rise in modern living standards makes the use of an electronic agenda indispensable.

- Storing daily operations and obligations in electronic form,
- Time management,
- Keeping informed about events like birthdays, appointments, celebrations

are some of the issues that can be solved by the electronic organizer's technology.

This diploma thesis deals with the creation and presentation of an electronic organizer. The application described below is implemented with the C# programming language in the Visual Studio platform.

Πίνακας περιεχομένων

Πίνακας εικόνων

Παραθέματα κώδικα

1. Εισαγωγή

Στο παρών έγγραφο αναλύεται και παρουσιάζεται η δημιουργία μιας ηλεκτρονικής ατζέντας. Πρόκειται για μια εφαρμογή που θα δίνει στον χρήστη τη δυνατότητα να οργανώσει τις επερχόμενες υποχρεώσεις του σε μηνιαίο και εβδομαδιαίο χρονοδιάγραμμα.

Η υλοποίηση της συγκεκριμένης εργασίας έχει ως βασικό σκοπό την εξοικείωση με τον τρόπο που οργανώνεται η περάτωση μιας εφαρμογής για ηλεκτρονικούς υπολογιστές. Η επιλογή της γλώσσας προγραμματισμού C# οδήγησε στην εκμάθηση των βασικών λειτουργιών που διαθέτει η γλώσσα σε συνδυασμό με το περιβάλλον ανάπτυξης εφαρμογών Visual Studio.

Κίνητρο για την διεξαγωγή της εργασίας είναι η γνωριμία με μια ολοκληρωμένη υλοποίηση εφαρμογής από το αρχικό στάδιο του σχεδιασμού ενός γενικού πλάνου έως την τελική παρουσίαση του έργου.

1.1 Ηλεκτρονική Ατζέντα

Η ανάπτυξη εφαρμογών για την υλοποίηση μιας ηλεκτρονικής ατζέντας προσφέρει στον χρήστη ένα εργαλείο καθημερινής χρήσης το οποίο συνδυάζει όλες τις απαραίτητες λειτουργίες για την οργάνωση των κοινωνικών και εργασιακών υποχρεώσεων.

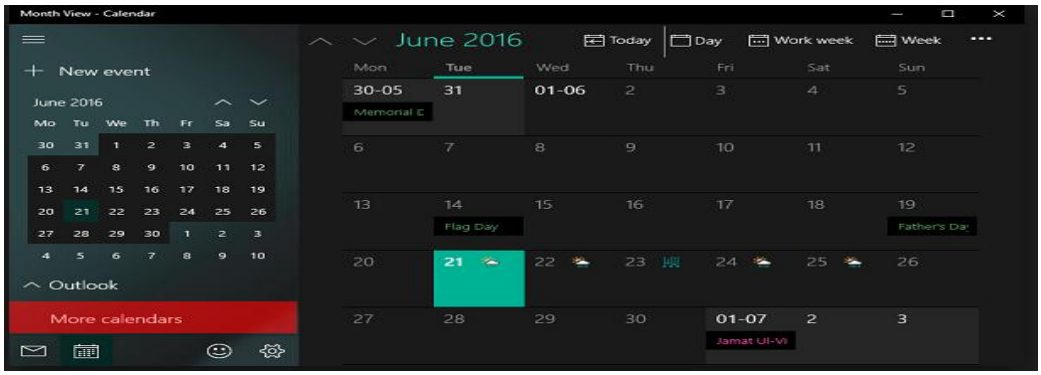
Μια ηλεκτρονική ατζέντα μπορεί να υλοποιηθεί σαν τοπική εφαρμογή για προσωπική χρήση όπως:

- η επέκταση Lightning από το Mazilla Thunderbird,
- το Microsoft Outlook χωρίς τον Exchange Server,
- το Windows Calendar

Ωστόσο μπορεί να υλοποιηθεί με διαδικτυακά πακέτα που θα επιτρέπουν την απομακρυσμένη διαμοίραση πληροφοριών μεταξύ των χρηστών. Τέτοιου είδους εφαρμογές είναι οι παρακάτω:

- Mozilla Sunbird
- Windows Live Calendar
- Google Calendar
- Microsoft Outlook με Exchange Server
- Apple Calendar

- Sunrise
- Digical



Εικόνες 1 Calendar [13]

1.2 Απαραίτητα χαρακτηριστικά

Οι εφαρμογές που υλοποιούν μια ηλεκτρονική ατζέντα θα πρέπει να περιλαμβάνουν κάποιες από τις παρακάτω λειτουργίες.

- Ημερολόγιο. Προβολή ετήσιου και εβδομαδιαίου χρονοδιαγράμματος.
- Λίστα επαφών.
- Δημιουργία γεγονότος.
- Προβολή των γεγονότων και υπενθύμιση.
- Κοινή χρήση και διαχείριση των γεγονότων με άλλους χρήστες.
- Πολλαπλές επιλογές στην προβολή του ημερολογίου και των γεγονότων.

1.3 Βασικές αρχές για εύχρηστη ατζέντα

Οι εφαρμογές ημερολογίου έχουν δύο βασικές λειτουργίες:

- Ø Εμφανίζουν τον προγραμματισμό των επερχόμενων γεγονότων και
- Ø Υπενθυμίζουν στον χρήστη τα σημαντικά γεγονότα.

Τα καλύτερα ημερολόγια, ωστόσο, δεν είναι απλώς λειτουργικά. Είναι διασκεδαστικά χρήσιμα και βοηθούν στην αύξηση της παραγωγικότητας των χρηστών.

Για την καλύτερη εμπειρία του χρήστη από την εφαρμογή θα πρέπει να ακολουθούνται τουλάχιστον οι παρακάτω αρχές:

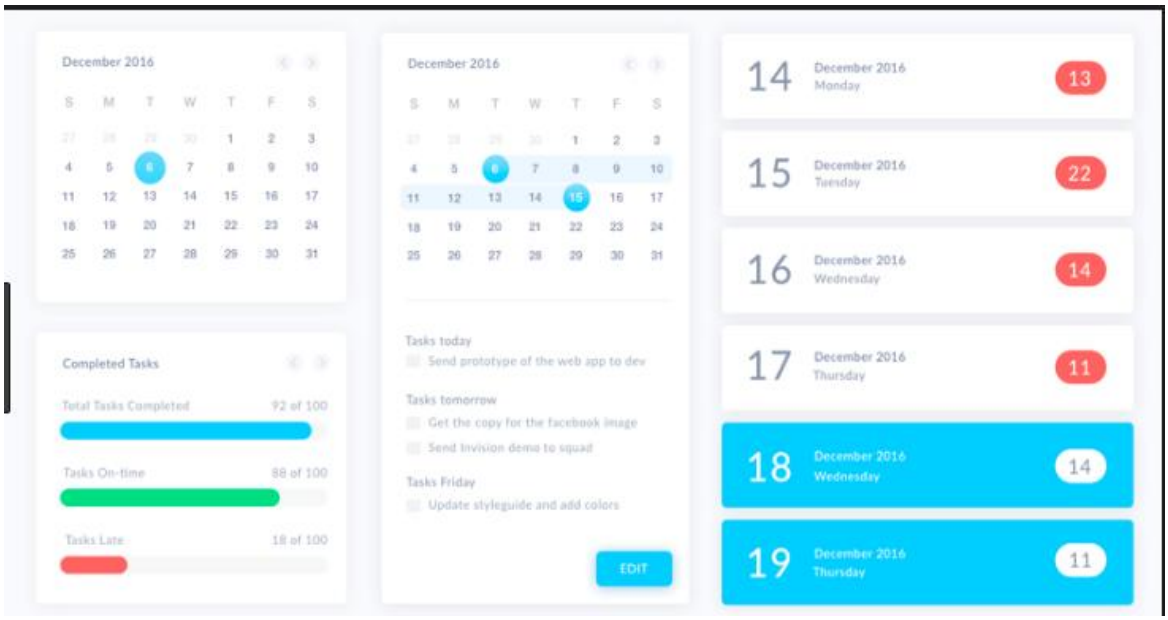
Ευκολία στη χρήση.

Εάν για παράδειγμα η προσθήκη ενός νέου γεγονότος απαιτεί αρκετές ενέργειες από τον χρήστη σε πραγματικό χρόνο εκτέλεσης της εφαρμογής είναι βέβαιο ότι θα του προκαλέσει δυσφορία και θα τον οδηγήσει στην αναζήτηση κάποιας άλλης εφαρμογής. Ακόμη η προβολή των επερχόμενων συμβάντων είναι μια διαδικασία που δεν πρέπει να απαιτεί εξερεύνηση σε διάφορα μενού αλλά να είναι εύκολο στην πρόσβαση από την αρχική σελίδα της εφαρμογής. [7]

Ευχάριστη αισθητική.

Ένα όμορφο , ξεκούραστο και ξεκάθαρο περιβάλλον χρήστη (UI) κάνει τον έλεγχο και την περιήγηση στην ατζέντα ευχάριστη διαδικασία. Η επίσκεψη στην ατζέντα από τον χρήστη πραγματοποιείται αρκετές φορές στην διάρκεια της ημέρας όποτε πρέπει να του προσφέρει ένα φιλόξενο περιβάλλον και μια ευχάριστη εμπειρία. Πέρα από την καλή εμφάνιση πολύ σημαντικός παράγοντας στην εμπειρία χρήστη είναι ο σχεδιασμός. Το πρόγραμμα πρέπει να σχεδιάζεται έτσι ώστε να μην αποπροσανατολίζει τον χρήστη και να τον κατατοπίζει εύκολα στα διαφορετικά εργαλεία που του προσφέρονται. [7]

Εικόνες 2 Κακό περιβάλλον χρήστη [14]



Εικόνες 3 Καλή αισθητική περιβάλλοντος [15]

Ευκολία στην κοινή χρήση πληροφοριών.

Βασική λειτουργία των σύγχρονων organizers είναι η από κοινού διαχείριση των γεγονότων με συνεργάτες και φίλους. Όπως επίσης η ανταλλαγή πληροφοριών είναι ενέργειες που πρέπει να προσφέρονται στον χρήστη με ευκολία.

Στο παράδειγμα του Google Calendar χρησιμοποιείται η τεχνολογία Cloud Computing. Πρόκειται για μια τεχνολογία που δίνει τη δυνατότητα στους χρήστης της να έχουν πρόσβαση σε εφαρμογές και συσκευές απομακρυσμένα. Προσφέρει υπηρεσίες διαμοιρασμού πόρων για την πρόσβαση και την αποθήκευση πληροφοριών χωρίς να βρίσκεται ο χρήστης στη συσκευή που διατηρεί τις πληροφορίες.

Εικόνες 4 Cloud Computing [16]

2. Τεχνολογίες που χρησιμοποιήθηκαν

2.1 Visual Studio

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE Integrated Drive Electronics) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη:

- προγραμμάτων ηλεκτρονικών υπολογιστών,
- ιστοσελίδων,
- εφαρμογών ιστού,
- υπηρεσιών ιστού και
- εφαρμογών για κινητά.

Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft όπως τα Windows API, τα Windows Forms, το Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Μπορεί να παράγει τόσο εγγενή κώδικα όσο και διαχειριζόμενο κώδικα.

Το Visual Studio περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα το οποίο υποστηρίζει το IntelliSense (το στοιχείο ολοκλήρωσης κώδικα) καθώς και τον refactoring κώδικα. Το ολοκληρωμένο πρόγραμμα εντοπισμού σφαλμάτων λειτουργεί τόσο σε επίπεδο πηγής όσο και ως εργαλείο εντοπισμού σφαλμάτων σε επίπεδο μηχανής. Άλλα ενσωματωμένα εργαλεία περιλαμβάνουν έναν προφίλ κώδικα, σχεδιαστή μορφών για την κατασκευή εφαρμογών GUI, σχεδιαστής ιστοσελίδων, σχεδιαστής τάξεων και σχεδιαστής σχήματος βάσης δεδομένων. Αποδέχεται plug-ins που βελτιώνουν τη λειτουργικότητα σχεδόν σε όλα τα επίπεδα συμπεριλαμβανομένης της προσθήκης υποστήριξης για συστήματα ελέγχου πηγής (όπως το Subversion και το Git) και την προσθήκη νέων εργαλείων όπως οι εκδότες και οι οπτικοί σχεδιαστές για συγκεκριμένες γλώσσες ή ομάδες εργαλείων για άλλες πτυχές της ανάπτυξης λογισμικού.

Το Visual Studio υποστηρίζει 36 διαφορετικές γλώσσες προγραμματισμού και επιτρέπει στον επεξεργαστή κειμένου και στον εντοπισμό σφαλμάτων να υποστηρίζει (σε διαφορετικούς βαθμούς) σχεδόν οποιαδήποτε γλώσσα προγραμματισμού, υπό την προϋπόθεση ότι υπάρχει.

Εικόνες 5 Visual Studio [4]

Οι ενσωματωμένες γλώσσες περιλαμβάνουν τις C, C ++, C ++ / CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML και CSS. Υποστηρίζει ακόμη γλώσσες όπως Python, Ruby, Node.js. Η Java (και J #) υποστηρίχθηκαν στο παρελθόν.[5]

2.2 C#

Η C# είναι μια σύγχρονη αντικειμενοστραφής γλώσσα προγραμματισμού, που αναπτύχθηκε το 2000 από τον Anders Hejlsberg στη Microsoft ως αντίπαλος της Java (στην οποία μοιάζει αρκετά). Δημιουργήθηκε λόγω ότι η Sun (που αργότερα αγόρασε η Oracle) δεν ήθελε να κάνει αλλαγές στη Java, έτσι η Microsoft επέλεξε να δημιουργήσει τη δική της γλώσσα. Η C# έχει αναπτυχθεί ραγδαία από τότε που δημιουργήθηκε για πρώτη φορά, με εκτεταμένη υποστήριξη από τη Microsoft που την οδήγησε να κερδίσει μια μεγάλη ανταπόκριση. Πλέον είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο.

Εικόνες 6 C# λογότυπο [17]

Πρόκειται για μια γλώσσα «γενικής χρήσης» που έχει σχεδιαστεί για την ανάπτυξη εφαρμογών στην πλατφόρμα της Microsoft και απαιτεί το .NET framework στα Windows προκειμένου να λειτουργήσει. Η C# συχνά θεωρείται ως ένα υβρίδιο που περιέχει τα πλεονεκτήματα από τη C και τη C++ για να δημιουργήσει μια πραγματικά εκσυγχρονισμένη γλώσσα. Παρόλο που το .NET Framework υποστηρίζει πολλές άλλες γλώσσες κωδικοποίησης, η C# έγινε γρήγορα μια από τις πιο δημοφιλείς.

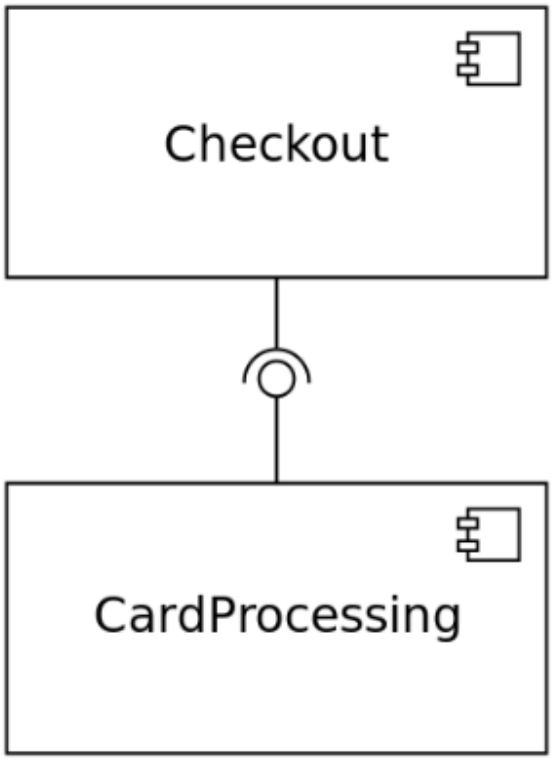
Η C# μπορεί να χρησιμοποιηθεί για να δημιουργήσει σχεδόν οτιδήποτε, αλλά είναι ιδιαίτερα ισχυρή στην κατασκευή εφαρμογών και παιχνιδιών για υπολογιστές Windows. Η C# μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών ιστού και έχει γίνει όλο και πιο δημοφιλές για την ανάπτυξη εφαρμογών κινητής τηλεφωνίας επίσης. Τα εργαλεία πολλαπλής πλατφόρμας όπως το Xamarin επιτρέπουν σε εφαρμογές γραμμένες σε C # να χρησιμοποιούνται σε σχεδόν οποιαδήποτε κινητή συσκευή. [2]

Οι κύριοι στόχοι που τέθηκαν κατά τη σχεδίαση της συγκεκριμένης γλώσσας συνοψίζονται στους παρακάτω:[10]

- Σχεδιάστηκε να είναι μια απλή, σύγχρονη, γενικής χρήσης, αντικειμενοστραφή γλώσσα προγραμματισμού.
- Η γλώσσα και οι εφαρμογές της πρέπει να παρέχουν υποστήριξη για τις αρχές δημιουργίας λογισμικού όπως ο έλεγχος ισχυρού

τύπου (αυστηροί κανόνες για τη σύνταξη κατά τη μεταγλώττιση), έλεγχος ορίων πινάκων, ανίχνευση προσπαθειών για χρήση μεταβλητών που δεν έχουν λάβει αρχική τιμή και αυτόματη συλλογή απορριμμάτων (εκκαθάριση των μη χρήσιμων πλέον δεδομένων).

- Η C# σχεδιάστηκε για χρήση στην ανάπτυξη software components κατάλληλων για ανάπτυξη σε κατανεμημένα περιβάλλοντα. Στην εικόνα 7 φαίνονται δυο components σε UML όπου το component της πληρωμής είναι υπεύθυνο για τη ολοκλήρωση της παραγγελίας του πελάτη και απαιτεί την πιστωτική κάρτα που παρέχει το δεύτερο component.[8]



Εικόνες 7 Παράδειγμα δυο components [8]

- Η φορητότητα των εφαρμογών είναι πολύ σημαντική για τον πηγαίο κώδικα και τους προγραμματιστές, ειδικά εκείνους που είναι ήδη εξοικειωμένοι με τη C και τη C ++.

- Η υποστήριξη της διεθνοποίησης. Της υιοθέτησης του λογισμικού από διαφορετικές γλώσσες και διαφορετικές απαιτήσεις συστήματος.[9]
- Η C# προορίζεται να είναι κατάλληλη για την δημιουργία εφαρμογών τόσο για φιλοξενούμενα όσο και για ενσωματωμένα συστήματα. Το εύρος που σχεδιάστηκε να καλύπτει είναι από πολύ μεγάλα συστήματα που χρησιμοποιούν εξελιγμένα λειτουργικά συστήματα μέχρι και τα πολύ μικρά που διαθέτουν συγκεκριμένες λειτουργίες.
- Παρόλο που οι εφαρμογές C# προορίζονται να είναι οικονομικές όσον αφορά τις απαιτήσεις ισχύος μνήμης και επεξεργασίας, η γλώσσα δεν προορίζεται να ανταγωνιστεί άμεσα στην απόδοση και το μέγεθος τη γλώσσα C ή τη assembly.

Ένα απλό πρόγραμμα σε C#.

Στο παράθεμα κώδικα που ακολουθεί υλοποιείται το κλασικό “Hello World” παράδειγμα.

```
1  using System;
2
3  class Program
4  {
5      static void Main()
6      {
7          Console.WriteLine("Hello, world!");
8      }
9  }
```

Κώδικας 1 Hello World στη C#.

Η εκτέλεση του παραπάνω θα εμφανίσει την πρόταση “Hello, world!” στην οθόνη του χρήστη.

Η πρώτη γραμμή κώδικα λέει στον μεταγλωττιστή να χρησιμοποιεί το “*System*” ως πρόθεμα για τύπους που χρησιμοποιούνται στον πηγαίο κώδικα. Σε αυτή την περίπτωση, όταν ο μεταγλωττιστής βλέπει τη χρήση του τύπου “*Console*” στον πηγαίο κώδικα, προσπαθεί να βρει έναν τύπο που

ονομάζεται *“Console”*, πρώτα στην τρέχουσα assembly και έπειτα σε όλες τις αναφερόμενες προς χρήση assembly αναφορές. Εδώ ο μεταγλωττιστής δεν θα βρει έναν τέτοιο τύπο, δεδομένου ότι το όνομα του τύπου είναι στην πραγματικότητα *“System.Console”*. Στη συνέχεια, ο μεταγλωττιστής επιχειρεί να βρει έναν τύπο που ονομάζεται *“System.Console”* χρησιμοποιώντας το πρόθεμα συστήματος από τη δήλωση *“using”* και αυτή τη φορά επιτυγχάνει. Η εντολή *“using”* επιτρέπει στον προγραμματιστή να δηλώσει όλα τα προθέματα που πρέπει να χρησιμοποιήσει κατά τη διάρκεια της μεταγλώττισης, αντί να χρησιμοποιεί πάντα ονόματα πλήρους τύπου.

Η γραμμή 3 είναι η δήλωση μιας νέας κλάσης και οτιδήποτε ακολουθεί μέσα στα άγκιστρα είναι η περιγραφή της κλάσης *“Program”*.

Η γραμμή 5 δηλώνει τη μέθοδο της κλάσης από όπου θα ξεκινάει η εκτέλεση του προγράμματος. Αφού γίνει έναρξη της εφαρμογής το .NET καλεί την *“Main”* μέθοδο. (Η κύρια μέθοδος, ωστόσο μπορεί να καλείται από οποιαδήποτε άλλη μέθοδο του προγράμματος). Η δεσμευμένη λέξη *“static”* καθιστά τη μέθοδο προσβάσιμη χωρίς τη δημιουργία στιγμιότυπου της κλάσης *“Program”*. Η κύρια μέθοδος (*“Main”)* κάθε εφαρμογής κονσόλας πρέπει να δηλώνεται στατική. Διαφορετικά, κάθε πρόγραμμα θα απαιτούσε ένα στιγμιότυπο της κλάσης, αλλά και κάθε στιγμιότυπο θα απαιτούσε ένα πρόγραμμα. Για να αποφευχθεί αυτή η ανεπανόρθωτη κυκλική εξάρτηση (circular dependency), οι μεταγλωττιστές της C# που επεξεργάζονται εφαρμογές κονσόλας (όπως η παραπάνω) αναφέρουν ένα σφάλμα αν δεν υπάρχει στατική κύρια μέθοδος. Η δεσμευμένη λέξη *“void”* δηλώνει ότι η *“Main”* δεν έχει τιμή επιστροφής.

Η εντολή που βρίσκεται μέσα στο σώμα της κύριας μεθόδου εμφανίζει την έξοδο του προγράμματος στην οθόνη. Η *“Console”* είναι μια στατική κλάση στο *namespace “System”*. Παρέχει μια διασύνδεση με τις τυπικές εισόδους, εξόδους και τις ροές σφαλμάτων για εφαρμογές κονσόλας. Το πρόγραμμα καλεί τη μέθοδο *“WriteLine”* της κλάσης *“Console”*, η οποία εμφανίζει στην οθόνη εκτέλεσης μια γραμμή με το όρισμα που δέχεται η μέθοδος, τη συμβολοσειρά "Hello World!".

2.3 Visual Studio και Windows Forms

Η C# με τη χρήση του Visual Studio υλοποιεί εφαρμογές όπως το Windows Form. Πρόκειται για μια συλλογή από στοιχεία ελέγχου όπως:

- Labels
- Text Boxes
- Combo Boxes
- Containers

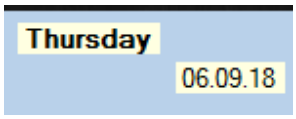
Label

Τα *labels* (ετικέτες) χρησιμοποιούνται ώστε να εμφανίσουν κείμενο στη φόρμα του προγράμματος. Είναι ένα στοιχείο ελέγχου που χρησιμοποιείται αρκετά συχνά στις εφαρμογές καθώς η επικοινωνία με τον χρήστη σε πραγματικό χρόνο (run-time) είναι σημαντική για μια καλή λειτουργία του προγράμματος. Στην εφαρμογή που περιγράφεται στο κεφάλαιο 3, τα *labels* έχουν χρησιμοποιηθεί ώστε:

Ø Να επεξηγούν στον χρήστη το ρόλο του text box που ακολουθεί,

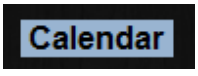
Εικόνες 8 Label εφαρμογης i

Ø να εμφανίζουν τις απαραίτητες πληροφορίες (παράδειγμα για την συγκεκριμένη ημερομηνία),



Εικόνες 9 Label εφαρμογής ii

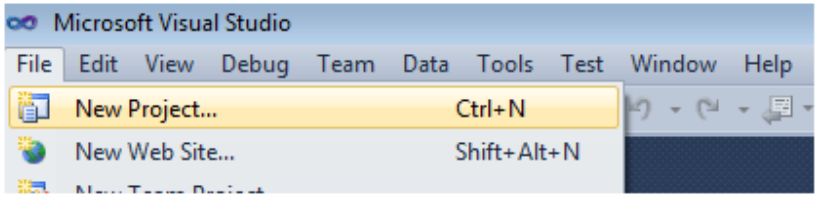
Ø να δηλώσουν το ρόλο της εφαρμογής.



Εικόνες 10 Lable εφαρμογής iii

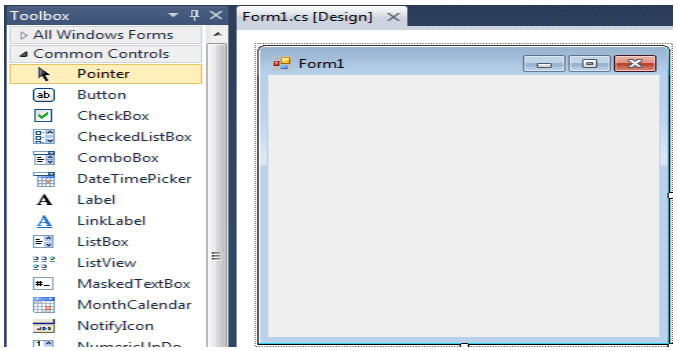
Η δημιουργία ενός *label* στο Visual Studio μπορεί να πραγματοποιηθεί με δυο τρόπους. Στο στάδιο της σχεδίασης (design-time) ή στο στάδιο της εκτέλεσης (run-time).

Το πρώτο βήμα είναι να ξεκινήσουμε ένα νέο έργο και να δημιουργήσουμε μια φόρμα (Windows Form Application). Μετά την εκκίνηση του Visual Studio επιλέγουμε File -> New Project και από το παράθυρο διαλόγου New Project επιλέγουμε Other Languages -> Visual C#. Έπειτα επιλέγουμε το Windows Forms Application. Καταχωρούμε ένα όνομα έργου στο κάτω μέρος του παραθύρου που εμφανίζεται και κάνουμε κλικ στο κουμπί OK. Η παρακάτω εικόνα δείχνει τα βήματα για την δημιουργία μιας νέας φόρμας στο Visual Studio.



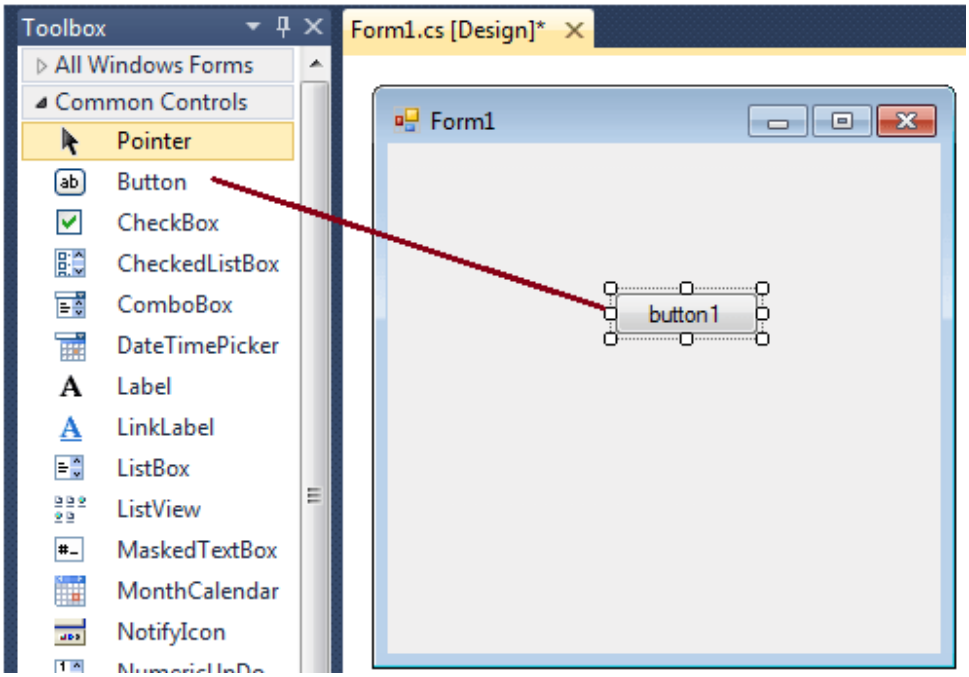
Εικόνες 11 Νεο project (a)

Αφού επιλέξουμε την εφαρμογή Windows Forms, μπορούμε να δούμε μια προεπιλεγμένη φόρμα (Form1) στο νέο project C#. Το έντυπο των Windows που φαίνεται στην προβολή σχεδιαστή είναι μια οπτική αναπαράσταση του παραθύρου που θα ανοίξει όταν ανοίξει η εφαρμογή. Η πλοήγηση μεταξύ αυτής της προβολής και της προβολής κώδικα γίνεται ανά πάσα στιγμή κάνοντας δεξί κλικ στο παράθυρο του σχεδίου ή του παραθύρου κώδικα και στη συνέχεια κάνοντας κλικ στην επιλογή Προβολή κώδικα ή Προβολή σχεδιαστή. Η παρακάτω εικόνα δείχνει πώς φαίνεται η προεπιλεγμένη φόρμα (Form1).



Εικόνες 12 Default Form

Κάθε φορά που δημιουργείται μια εφαρμογή των Windows, το Visual Studio θα εμφανίσει μια προεπιλεγμένη κενή φόρμα στην οποία μας επιτρέπει να σύρουμε τα στοιχεία ελέγχου στην κύρια φόρμα εφαρμογών με το κατάλληλο μέγεθος και τη σωστή θέση.

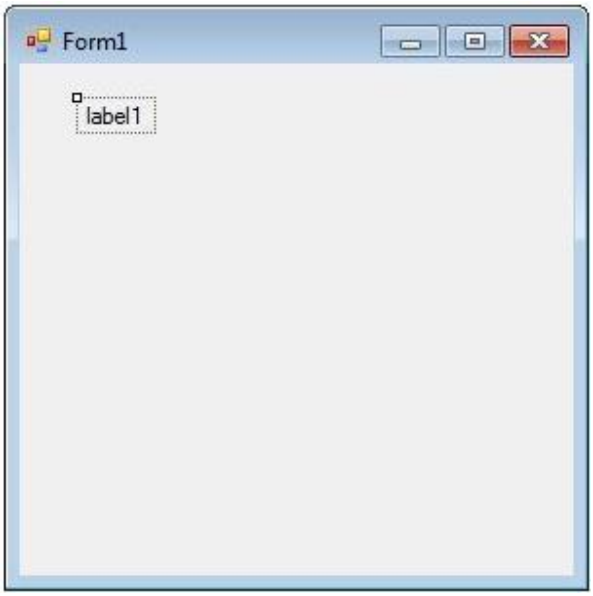


Εικόνες 13 Visual Studio WinForm

Design-time

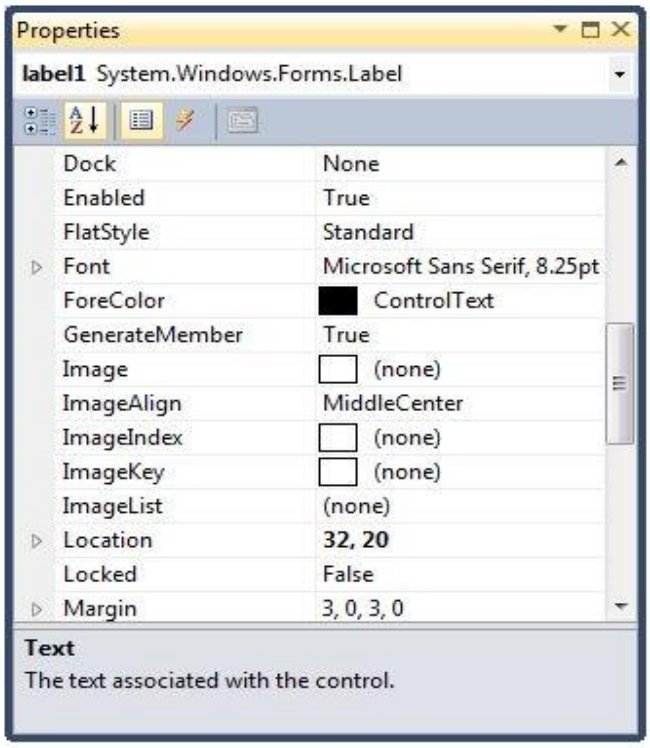
Αρχικά, μπορούμε να χρησιμοποιήσουμε τον σχεδιαστή φόρμας του Visual Studio για να δημιουργήσουμε έναν έλεγχο κατά το σχεδιασμό. Στη λειτουργία του Design-time, μπορούμε να χρησιμοποιήσουμε οπτικές διεπαφές χρήστη για να δημιουργήσουμε ιδιότητες ελέγχου και να γράψουμε μεθόδους.

Για τη δημιουργία ενός στοιχείου ετικέτας κατά το σχεδιασμό, απλώς σύρουμε ένα στοιχείο ετικέτας από την εργαλειοθήκη (toolbox) του Visual Studio και το κάνουμε απόθεση στη φόρμα (drag-n-drop). Αφού ολοκληρώσουμε την παραπάνω λειτουργία η φόρμα θα είναι όπως αυτή στην εικόνα 14.



Εικόνες 14 Δημιουργία Label σε Design-time

Έπειτα μπορούμε να μετακινήσουμε την ετικέτα και να της αλλάξετε μέγεθος χρησιμοποιώντας το ποντίκι. Επόμενο βήμα είναι να ορίσουμε τις ιδιότητες από την καρτέλα Properties που παρέχει το Visual Studio(εικόνα 15).



Εικόνες 15 Properties

Run-time

Η κλάση Label αφορά ένα Label control. Δημιουργούμε ένα νέο στιγμιότυπο (instance) της κλάσης Label έπειτα θέτουμε τις ιδιότητες του και το προσθέτουμε στη φόρμα.

1. Το πρώτο βήμα είναι να δημιουργήσουμε ένα νέο στιγμιότυπο της κλάσης Label. Το ακόλουθο κομμάτι κώδικα κατασκευάζει ένα νέο αντικείμενο της συγκεκριμένης κλάσης.

```
// Create a Label object
Label dynamicLabel = new Label();
```

Κώδικας 2 Δημιουργία Label αντικειμένου

2. Στο επόμενο βήμα θέτουμε τις ιδιότητες της ετικέτας. Το παράθεμα κώδικα που ακολουθεί θέτει:
   - το χρώμα που θα φαίνεται στο παρασκήνιο (background) της ετικέτας,
   - το χρώμα που θα εμφανίζεται στο προσκήνιο (foreground),
   - το κείμενο που θα έχει η ετικέτα
   - το όνομα του αντικειμένου
   - τη γραμματοσειρά και το μέγεθος των γραμμάτων

```
dynamicLabel.BackColor = Color.Red;
dynamicLabel.ForeColor = Color.Blue;

dynamicLabel.Text = "I am a Dynamic Label";
dynamicLabel.Name = "DynamicLabel";
dynamicLabel.Font = new Font("Georgia", 16);
```

Κώδικας 3 Ιδιότητες του Label

3. Τέλος είναι απαραίτητο να εισάγουμε τη νέα ετικέτα που δημιουργήσαμε στη φόρμα, χρησιμοποιώντας τη μέθοδο *Form.Controls.Add*.

```
Controls.Add(dynamicLabel);
```

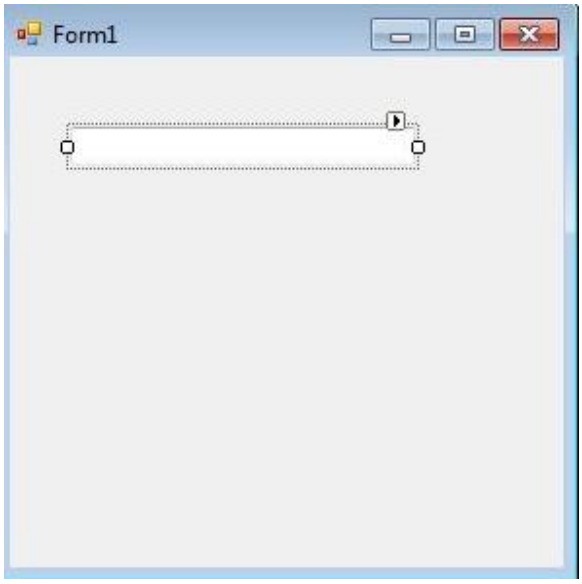
Κώδικας 4 Εισαγωγή του Label

Text Box

Ένα ακόμη στοιχείο ελέγχου που είναι απαραίτητο σε διαδραστικές εφαρμογές είναι το Text Box. Πρόκειται για ένα εργαλείο που δέχεται την είσοδο χρήστη σε μια φόρμα. Το Text Box μαζί με τα Labels είναι τα μέσα επικοινωνίας (είσοδος, έξοδος) με τον χρήστη της εφαρμογής.

Για την δημιουργία ενός νέου Text Box στη φόρμα το Visual Studio και η C# παρέχουν δυο διαφορετικούς τρόπους. Στο στάδιο της σχεδίασης ή με την παραγωγή κώδικα σε run-time. Η διαδικασία είναι παρόμοια με εκείνη των Labels.

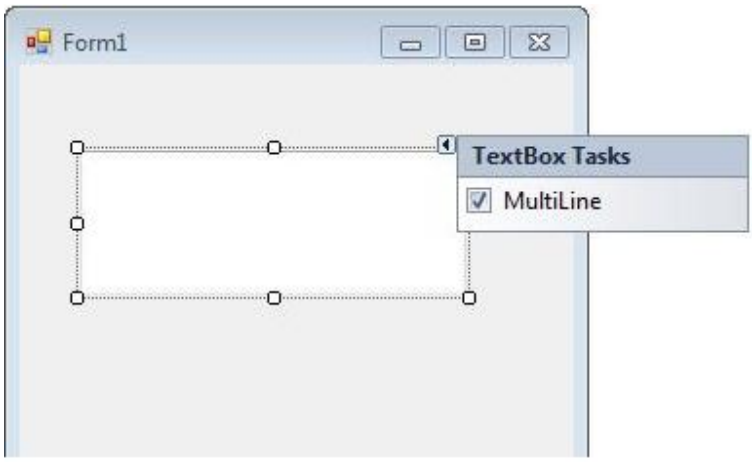
Μια ιδιότητα (Property) των TextBoxes είναι η επιλογή αν θα επιτρέπει μια γραμμή εμφάνισης κειμένου ή πολλαπλες. Με την εισαγωγή ενός νέου Text Box στη φόρμα, η προεπιλεγμένη κατάσταση είναι η εμφάνιση του κειμένου σε μία γραμμή όπως φαίνεται στην εικόνα 16.



Εικόνες 16 Text Box με μια γραμμή

Μπορούμε να αλλάξουμε την συγκεκριμένη επιλογή αν κάνουμε κλικ στη διαχείριση εργασιών (Task handle, είναι το βέλος πάνω από το Text Box) και επιλέξουμε το Check Box Multiline. Τώρα το Text Box έχει τη δυνατότητα να μεγαλώσει το ύψος ανάλογα τις απαιτήσεις της εφαρμογής.

Ένα παράδειγμα Text Box πολλαπλών γραμμών φαίνεται στην παρακάτω εικόνα.



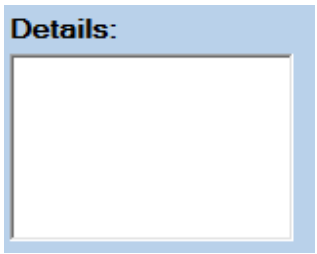
Εικόνες 17 Text Box MultiLine

Στην εφαρμογή που υλοποιήθηκε στα πλαίσια της συγκεκριμένης πτυχιακής το εργαλείο Text Box φάνηκε χρήσιμο για τις παρακάτω λειτουργίες:

1. Είσοδος των δεδομένων από τον χρήστη για την εισαγωγή νέου γεγονότος:
    1. Τίτλος. SingleLine.

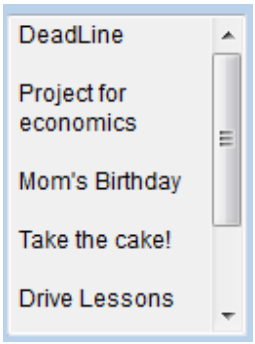
Εικόνες 18 Text Box για τον τίτλο γεγονότος

    2. Λεπτομέρειες του γεγονότος. MultiLine.



Εικόνες 19 Text Box για την περιγραφή γεγονότος

2. Εμφάνιση των γεγονότων της επιλεγμένης ημέρας. MultiLine με τη δυνατότητα κάθετης μπάρας κύλισης (scroll bar).



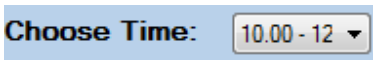
Εικόνες 20 Text Box για τα γεγονότα της ημέρας

Combo Box

Το εργαλείο έλεγχου ComboBox συνδυάζει τη λειτουργικότητα ενός πλαισίου κειμένου και ενός πλαισίου λίστας. Μόνο ένα στοιχείο από τη λίστας εμφανίζεται κάθε στιγμή σε ένα ComboBox ενώ οι υπόλοιπες διαθέσιμες επιλογές φορτώνονται σε μια αναπτυσσόμενη λίστα.

Για την κατασκευή ενός Combo Box χρειάζεται παρόμοια διαδικασία με του Label και του Text Box.

Στην δική μας εφαρμογή το Combo Box έδωσε λύση αισθητικού και λειτουργικού περιεχομένου στο πρόβλημα της επιλογής ώρας για το γεγονός που καταχωρεί ο χρήστης.



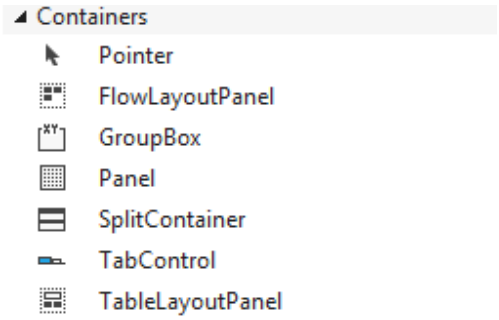
Εικόνες 21 Combo Box για την επιλογή ώρας

Containers

Ένας Container είναι ένα εξειδικευμένο αντικείμενο συλλογής που παρέχει λογική ομαδοποίηση για ένα ή περισσότερα στοιχεία. Οι Containers διαχειρίζονται την αλληλεπίδραση των στοιχείων τόσο μεταξύ τους όσο και με το εξωτερικό περιβάλλον της εφαρμογής. Η χρήση των Container μας επιτρέπει την διαχείριση των εξαρτημάτων που περιέχει σαν ένα ευρετήριο. Μπορούμε να εντοπίσουμε και να μεταβάλλουμε κάποιο από τα στοιχεία

που περιέχει αναζητώντας τη θέση τους μέσα στον Container. Παρέχει επίσης ένα μέσο για την κοινή απόρριψη των εξαρτημάτων που έχουν τοποθετηθεί μέσα στον Container όταν οι ανάγκες της εφαρμογής απαιτεί να μην εμφανίζονται στη φόρμα. Η ομαδοποίηση των αντικειμένων που τοποθετούνται σε έναν Container αφορά περισσότερο λογική σύνδεση και όχι οπτική ή φυσική. [11]

Το Visual Studio παρέχει μια συλλογή από Containers στο ToolBox όπως φαίνεται στην εικόνα 22.



Εικόνες 22 Toolbox Containers

Από την παραπάνω συλλογή ,για την εφαρμογή που πραγματεύεται η συγκεκριμένη αναφορά επιλέχθηκαν τα:

1. TabControl προκειμένου να δίνεται η δυνατότητα εναλλαγής της εμφάνισης μεταξύ του μήνα και της εβδομάδας.
2. Panel.
    i. Για την εισαγωγή γεγονότος (addEventPanel).
    ii. Για την εμφάνιση της ημέρας (dayPanel).
    iii. Για την εμφάνιση του μήνα μέσα στη πρώτη καρτέλα του TabControl (calendarPanel).
    iv. Για την εμφάνιση της εβδομάδας μέσα στη δεύτερη καρτέλα του TabControl (weekPanel).

2.4 Διαγράμματα ροής δεδομένων

Ένα διάγραμμα ροής δεδομένων (DFD) καταγράφει τη ροή πληροφοριών για οποιαδήποτε διαδικασία ή σύστημα. Χρησιμοποιεί καθορισμένα σύμβολα όπως:

- ορθογώνια,
- κύκλους
- βέλη, και
- ετικέτες σύντομων κειμένων,

για την εμφάνιση δεδομένων εισόδου-εξόδου, σημείων αποθήκευσης και των διαδρομών μεταξύ κάθε προορισμού.

Τα διαγράμματα δεδομένων μπορούν να κυμαίνονται από απλές αναλύσεις διαδικασιών που έχουν σχεδιαστεί με το χέρι μέχρι ένα λεπτομερές DFD πολλαπλών επιπέδων που εμβαθύνουν προοδευτικά στον τρόπο με τον οποίο χειρίζονται τα δεδομένα. Μπορούν να χρησιμοποιηθούν για την ανάλυση ενός υπάρχοντος συστήματος ή ενός νέου μοντέλου. Όπως όλα τα διαγράμματα και γραφήματα που ακολουθούν τις βασικές αρχές σχεδίασης, ένα DFD μπορεί συχνά να λέει πράγματα που είναι δύσκολο να εξηγηθούν με λέξεις. Εξυπηρετούν ανάγκες περιγραφής ενός συστήματος και για το τεχνικό δυναμικό που απασχολείται στη υλοποίηση του συστήματος αλλά και για μη τεχνικό, από τον προγραμματιστή του έργου έως τον διευθύνοντα σύμβουλο.

Αυτός είναι ο λόγος για τον οποίο τα DFD παραμένουν τόσο δημοφιλή μετά από αρκετά χρόνια που εμφανίστηκαν. Παρόλο που λειτουργούν ικανοποιητικά για τη περιγραφή ροής δεδομένων ενός λογισμικού, είναι λιγότερο εφαρμόσιμα σήμερα στην οπτικοποίηση διαδραστικών λογισμικών ή συστημάτων προσανατολισμένων σε βάσεις δεδομένων ή σε πραγματικό χρόνο.

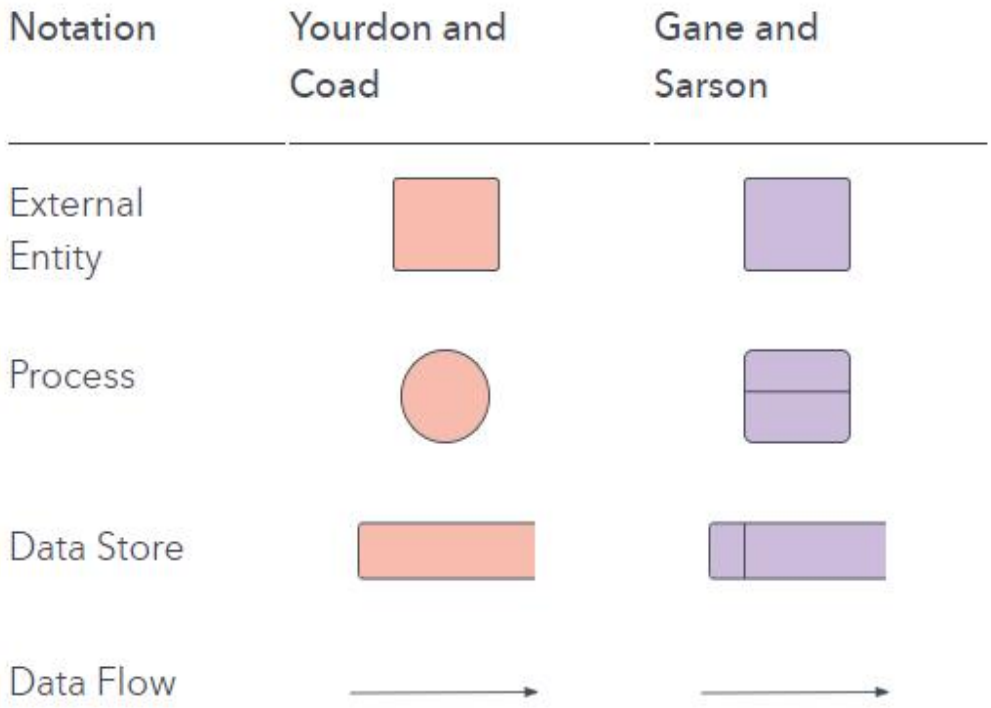
2.4.1 Σύμβολα και σημάνσεις

Στα διαγράμματα ροής δεδομένων υπάρχουν δυο κύρια συστήματα συμβόλων τα οποία πήραν το όνομα των δημιουργών τους:

- Yourdon και Coad
- Yourdon και DeMarco

- Gane και Sarson

Μια βασική διαφορά στα σύμβολά τους είναι ότι οι Yourdon-Coad και Yourdon-DeMarco χρησιμοποιούν κύκλους για τις διαδικασίες, ενώ οι Gane και Sarson χρησιμοποιούν ορθογώνια με στρογγυλεμένες γωνίες, που μερικές φορές ονομάζονται ρόμβοι. Επειδή υπάρχουν και άλλες παραλλαγές συμβόλων που χρησιμοποιούνται, είναι σημαντικό να είμαστε σαφείς και συνεπείς στα σχήματα και τις σημειώσεις που χρησιμοποιούμε προκειμένου να έχει νόημα η επικοινωνία με εταίρους συνεργάτες.



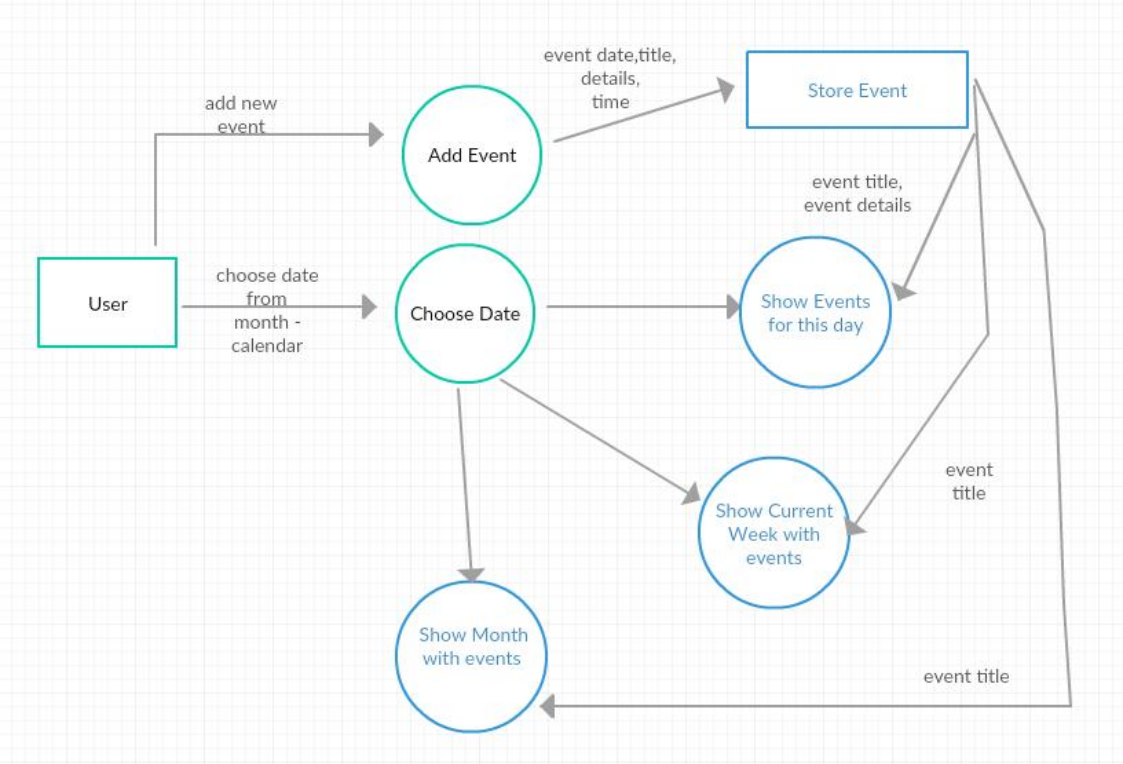
Εικόνες 23 Σύμβολα DFD [12]

Χρησιμοποιώντας κανόνες ή κατευθυντήριες γραμμές DFD οποιασδήποτε σύμβασης, τα σύμβολα απεικονίζουν τα τέσσερα στοιχεία των διαγραμμάτων ροής δεδομένων. Τα σύμβολα που εμφανίζονται στην εικόνα 23 περιγράφονται παρακάτω:

1. **Εξωτερική οντότητα (external entity)**: Είναι ένα εξωτερικό σύστημα που στέλνει ή λαμβάνει δεδομένα, επικοινωνώντας με το σύστημα που αφορά το διάγραμμα. Αυτές είναι οι πηγές και οι προορισμοί των πληροφοριών που εισέρχονται ή εξέρχονται από

το σύστημα. Μπορεί να είναι ένας εξωτερικός οργανισμός ή πρόσωπο, ένα σύστημα υπολογιστή ή ένα επιχειρηματικό σύστημα. Είναι επίσης γνωστοί ως terminators, πηγές, sinks ή ηθοποιοί. Συνήθως σχεδιάζονται στις άκρες του διαγράμματος.

2. **Διαδικασία (process)**: Οποιαδήποτε διαδικασία που αλλάζει τα δεδομένα, δημιουργώντας μια έξοδο. Μπορεί να εκτελέσει υπολογισμούς ή να ταξινομήσει δεδομένα με βάση τη λογική ή να κατευθύνει τη ροή δεδομένων με βάση τους επιχειρηματικούς κανόνες. Μια σύντομη ετικέτα χρησιμοποιείται για να περιγράψει τη διαδικασία, π.χ. "Υποβολή πληρωμής".
3. **Αποθήκη δεδομένων (data store)**: αρχεία ή αποθετήρια που περιέχουν πληροφορίες για μελλοντική χρήση, όπως ένας πίνακας βάσης δεδομένων ή μια φόρμα εγγραφής μέλους. Κάθε δομή δεδομένων περιγράφεται από μια ετικέτα, π.χ. "Παραγγελίες".
4. **Ροή δεδομένων (data flow):** Η διαδρομή που ακολουθούν τα δεδομένα μεταξύ των εξωτερικών οντοτήτων, των διαδικασιών και των αποθηκευμένων δεδομένων. Εμφανίζει τη διεπαφή μεταξύ των άλλων εξαρτημάτων και παρουσιάζεται με βέλη. Συνήθως έχει μια μικρή ετικέτα για το όνομα των δεδομένων, π.χ. "Στοιχεία χρέωσης". [12]



Εικόνες 24 DFD της εφαρμογής

Στην εικόνα 24 παρουσιάζεται το διάγραμμα ροής δεδομένων της εφαρμογής που αναλύεται στο 3ο κεφάλαιο. Ο χρήστης είναι η εξωτερική οντότητα του συστήματος. Οι δυο ενέργειες που εκτελεί είναι η εισαγωγή νέου γεγονότος και η επιλογή ημερομηνίας από το μηνιαίο ημερολόγιο που εμφανίζεται στο αριστερό μέρος της εφαρμογής.

Η αποθήκευση των νέων γεγονότων γίνεται στην λίστα *eventsData* και από εκεί ανακτώνται κάθε φορά που χρειάζεται να εμφανιστούν στα πλαίσιο της μεμονωμένης ημέρας, του μήνα και της εβδομάδας.

Η επιλογή μιας νέα ημερομηνίας από τον χρήστη της εφαρμογής προκαλεί αλλαγή στα δεδομένα που εμφανίζονται στην εβδομάδα, το μήνα και την ημέρα.

2.5 Σχεδιασμός και οργάνωση της εργασίας

Με τις παραπάνω δυνατότητες που προσφέρει ο συνδυασμός του Visual Studio και της C# ολοκληρώθηκε το μεγαλύτερο μέρος της εργασίας. Προηγήθηκε ο σχεδιασμός του πρότυπου παραθύρου σε χαρτί. Η επιλογή

των εργαλείων που χρησιμοποιήθηκαν ήταν σύμφωνα με τη συλλογή των εργαλείων ελέγχου που παρέχει το Windows Form Application.

Αφού σχεδιάστηκε η αρχική εμφάνιση της φόρμας της εφαρμογής ακολούθησε η επέμβαση στον κώδικα. Πρόκειται για το πιο απαιτητικό κομμάτι της εργασίας καθώς συναντήθηκαν ζητήματα προγραμματιστικής λογικής. Ένα παράδειγμα τέτοιου ζητήματος ήταν η αποθήκευση των δεδομένων που εισάγονται από το χρήστη και η εμφάνιση τους στο κατάλληλο σημείο της φόρμας όταν αυτό ζητηθεί.

Αφού ολοκληρώθηκε η αλγοριθμική επέμβαση επόμενο βήμα ήταν η ενασχόληση με την εμφάνιση και την αισθητική της εφαρμογής. Στο συγκεκριμένο στάδιο της εργασίας πάρθηκαν αποφάσεις για θέματα όπως:

- Τα χρώματα που θα πλαισιώνουν την εφαρμογή
- Τα σχήματα των κουμπιών ελέγχου και των λοιπών στοιχείων
- Το background της φόρμας
- Οι γραμματοσειρές

3 Παρουσίαση της εφαρμογής

Η εφαρμογή που υλοποιήθηκε αφορά ένα ηλεκτρονικό organizer με τα εξής χαρακτηριστικά:

- Προβολή μικρογραφίας μηνιαίου ημερολογίου με τη δυνατότητα πλοήγησης στους μήνες.

- Προβολή του μήνα που έχει επιλεχθεί. Εμφανίζονται σε μεγάλο εύρος του παραθύρου όλες οι μέρες του μήνα.
  - Για κάθε μέρα όπου έχει καταχωρηθεί γεγονός εμφανίζεται ο τίτλος του.
  - Η ημέρα που έχει επιλεχθεί από το μικρό ημερολόγιο εμφανίζεται με διακριτικό χρώμα.
  - Δίνεται η δυνατότητα στον χρήστη να προχωρήσει μπροστά και πίσω τους μήνες με δύο κουμπιά ελέγχου στο αριστερό και δεξί μέρος του μηνιαίου ημερολογίου.

- Δυνατότητα εισαγωγής νέου γεγονότος.

- Ο χρήστης μπορεί να επιλέξει την ημέρα που θέλει να εισάγει το γεγονός είτε από το μικρό ημερολόγιο που αναφέρθηκε στην αρχή είτε από τον drop down μενού που βρίσκεται στο πλαίσιο εισαγωγής γεγονότος.
- Δίνεται η δυνατότητα εισαγωγής τίτλου του γεγονότος.
- Εισαγωγή περιγραφής για το γεγονός.
- Επιλογή της ώρας με τη βοήθεια του combo box.

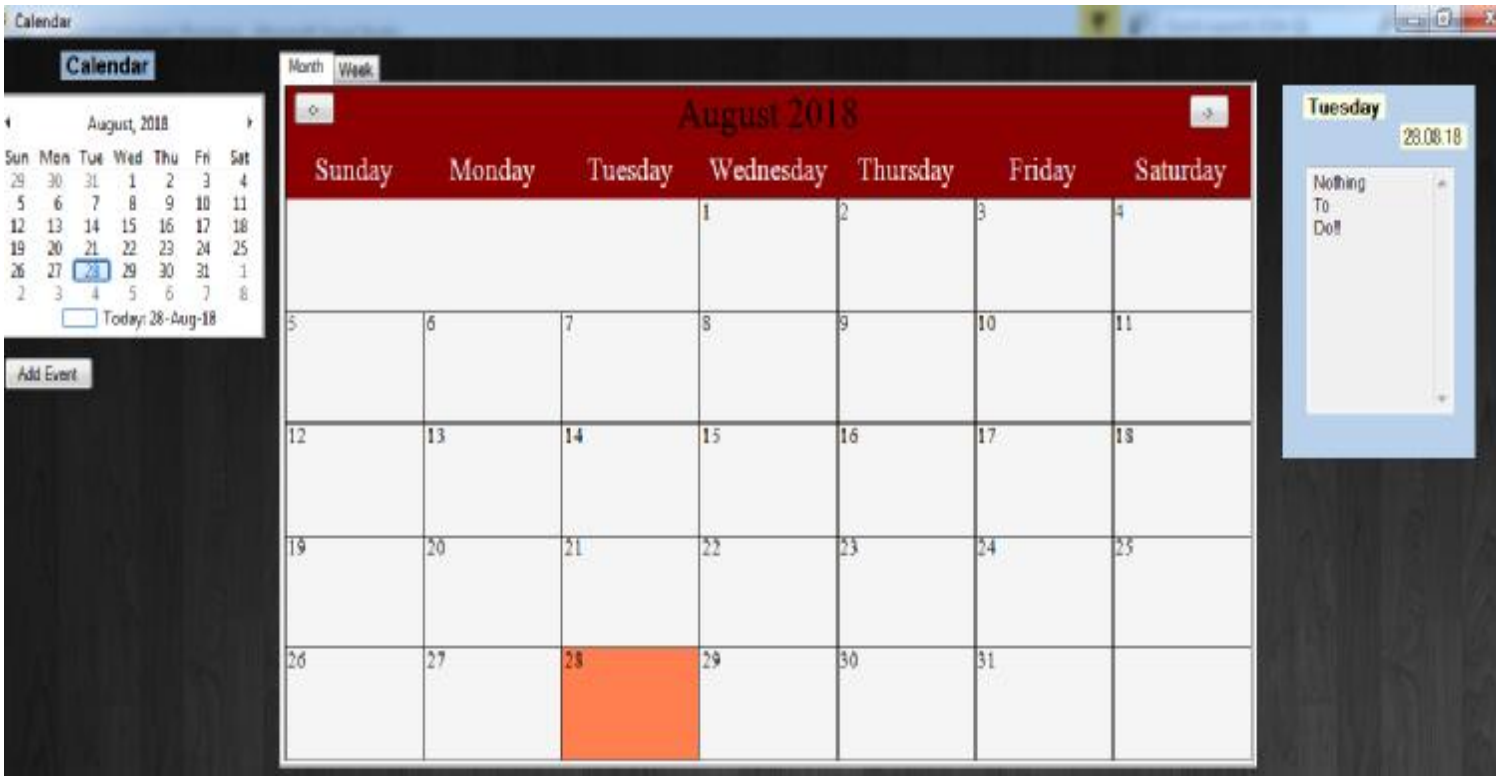
Ø Προβολή μιας ημέρα. Στο συγκεκριμένο πλαίσιο εμφανίζεται η επιλεγμένη ημέρα με τους τίτλους των γεγονότων, τις ώρες και τις περιγραφές τους.

Ø Προβολή εβδομαδιαίας κατάστασης. Η επιλεγμένη εβδομάδα από το ημερολόγιο εμφανίζεται σε ένα πλαίσιο με:
- την κάθε μέρα της εβδομάδας,
- τις ώρες που περιέχει το combo box,
- τα γεγονότα στην κατάλληλη ώρα και ημέρα.
- Δίνεται η δυνατότητα πλοήγησης στις εβδομάδες με επιλογή της επιθυμητής εβδομάδας από το ημερολόγιο που εμφανίζεται στα αριστερά της εφαρμογής.

3.1 Οργάνωση της φόρμας

Για την καλύτερη διαχείριση και οργάνωση της υλοποίησης ακολουθήθηκε μια βασική αρχή του προγραμματισμού «διαίρε και βασίλευε». Η ίδια τακτική είναι χρήσιμη και στην σχεδίαση της εφαρμογής.

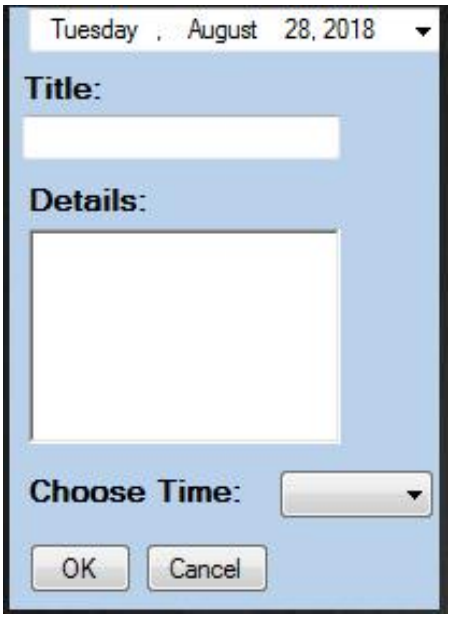
Αφού εκτελεστεί η εφαρμογή εμφανίζεται το παρακάτω Windows Form.



Εικόνες 25 Η αρχική εμφάνιση της εφαρμογής

Στην εικόνα που φαίνεται παραπάνω εμφανίζονται τέσσερα διαφορετικά πλαίσια για τις εκάστοτε λειτουργίες της εφαρμογής.

1) Πάνω αριστερά εμφανίζεται η μικρογραφία του ημερολογίου. Πρόκειται για την επιλογή MonthCalendar που προσφέρει η συλλογή της C# στο Visual Studio.
2) Κάτω αριστερά έχει τοποθετηθεί ένα κουμπί ελέγχου. Όταν ο χρήστης κάνει κλικ στο Add Event κουμπί εμφανίζεται ένα Panel. Για την καλύτερη ομαδοποίηση των στοιχείων ελέγχου της εφαρμογής το Windows Form Application παρέχει τους Containers. Εδώ γίνεται η επιλογή του Panel.



Εικόνες 26 Panel add Event

Όταν επιλέξει ο χρήστης την εισαγωγή γεγονότος εμφανίζεται το παραπάνω πλαίσιο στο κάτω αριστερό σημείου του παραθύρου.

3) Το μεγαλύτερο μέρος του παραθύρου το καταλαμβάνει η εμφάνιση του μήνα. Πρόκειται για έναν Container το TabControl που μας επιτρέπει να εισάγουμε δυο ή παραπάνω Containers σε κάθε καρτέλα του. Εδώ χρησιμοποιείται για να δίνεται στον χρήστη η δυνατότητα να προβάλει την μορφή του μήνα και της εβδομάδας όταν το επιθυμεί. Για την προβολή του μήνα γίνεται η χρήση ενός Panel και στην δεύτερη καρτέλα γίνεται η χρήση του DataGridView για την προβολή της εβδομάδας (εικόνα 11).

4) Στο δεξί μέρος της εφαρμογής εμφανίζεται η παρουσίαση της ημέρα που έχει επιλεχθεί. Επιλέχθηκε το Panel Container για την ομαδοποίηση των labels και των text fields που απαιτούνται.

Εικόνες 27 Εμφάνιση εβδομάδας και εισαγωγής γεγονότος

3.2 Ανάλυση του κώδικα

Στην ενότητα που ακολουθεί αναλύονται τα σημαντικότερα σημεία του κώδικα της εργασίας. Τα κομμάτια που εξετάζονται και παραπείθονται παρακάτω είναι σύμφωνα με την κατηγοριοποίηση που αναφέρθηκε στην προηγούμενη ενότητα.

3.2.1 Month Calendar

Στα εργαλεία που προσφέρει το Visual Studio για τη C# περιλαμβάνεται η προβολή ενός μηνιαίου ημερολογίου. Για να εισαχθεί στη φόρμα το ημερολόγιο αρκεί να το επιλέξουμε από τη λίστα του Toolbox -η καρτέλα στα αριστερά συνήθως στο Visual Studio- και να το τοποθετήσουμε σε όποιο σημείο της φόρμας επιθυμούμε (drag and drop). Η

εισαγωγή ενός νέου στοιχείο στο παράθυρο παράγει αυτόματα τον αντίστοιχο κώδικα στο αρχείο Form1.Designer.cs. Πρόκειται για τον κώδικα δήλωσης και αρχικοποίησης του στοιχείου.

```
// monthCalendar1
//
            this.monthCalendar1.Location = new System.Drawing.Point(3, 37);
            this.monthCalendar1.Name = "monthCalendar1";
            this.monthCalendar1.TabIndex = 0;
            this.monthCalendar1.DateChanged += new
System.Windows.Forms.DateRangeEventHandler(this.monthCalendar1_DateChanged);
//
```

Κώδικας 5 Month Calendar

Στην καρτέλα Form1.cs[Design] αφού εισαχθεί το νέο στοιχείο με διπλό κλικ πάνω του δημιουργείται μια μέθοδος στο Form1.cs. Πρόκειται για μια κενή αρχικά συνάρτηση όπου εισάγουμε τον κώδικα για τις ενέργειες που επιθυμούμε να εκτελούνται όταν πραγματοποιείται μια αλλαγή στο στοιχείο. Στο συγκεκριμένο «αντικείμενο» η αλλαγή που μπορεί να πραγματοποιηθεί είναι όταν αλλάζει ημερομηνία ο χρήστης.

Κατά την αλλαγή ημερομηνίας το πρόγραμμα απαιτεί να γίνουν και αλλαγές στα υπόλοιπα αντικείμενα της φόρμας.

- Στο κεντρικό ημερολόγιο πρέπει να εμφανίζεται ο αντίστοιχος μήνας.
- Στην προβολή της εβδομάδας πρέπει να εμφανίζεται η εβδομάδα που περιέχει την συγκεκριμένη ημέρα που επέλεξε ο χρήστης.
- Στην προβολή της ημέρας πρέπει να εμφανιστεί η συγκεκριμένη ημέρα και τα γεγονότα που έχουν αποθηκευτεί.

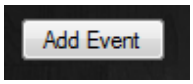
Στο απόσπασμα που ακολουθεί εμφανίζονται οι ενέργειες που περιεγράφηκαν. Η μεταβλητή FirstOfMonth είναι ένα καθολικό (global) στιγμιότυπο της κλάσης DateTime που προσφέρεται από τη C#. Καθορίζει τις ημερομηνίες που εμφανίζουν τα στοιχεία της φόρμας.

```
//Actions when selected date changed
private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)
{
            DateTime d = monthCalendar1.SelectionRange.Start;
            int year_num = d.Year;
            int month_num = d.Month;
            DateTime first_of_month =
                new DateTime(year_num, month_num, 1);

            weekPanel_initComp();
            dayPanel_initComp();

            FirstOfMonth = first_of_month;

            //set date time picker from the selected date
            dateTimePicker1.Value = d;
            //repaint week and calendar panels
            calendarPanel.Invalidate();
            weekPanel.Invalidate();
 }
```

Κώδικας 6 Action Listener Month Calendar

3.2.2 Add Event Panel

Ένα από τα ζητήματα που έπρεπε να επιλυθεί ήταν η εμφάνιση του Panel εισαγωγής νέου γεγονότος. Η μέθοδος που επιλέχθηκε είναι να εμφανίζεται ένα κουμπί ώστε όταν το επιλέγει ο χρήστης να προβάλετε το Panel εισαγωγής.



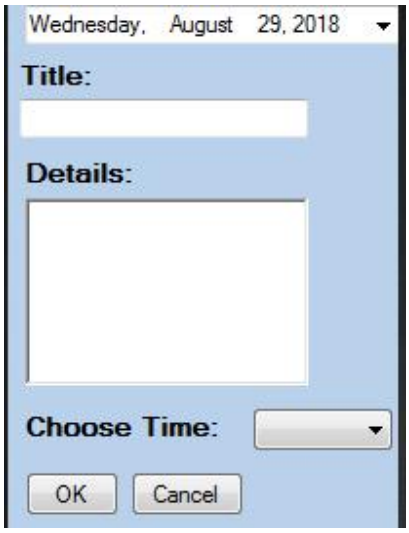
Εικόνες 28 Κουμπί εισαγωγής γεγονότος

Τα αντικείμενα της C# διαθέτουν αρκετές έτοιμες μεθόδους. Στη συγκεκριμένη περίπτωση η μέθοδος Visible εξυπηρετεί το σκοπό μας καθώς μας δίνει τη δυνατότητα να εμφανίζουμε ή όχι το Panel της εισαγωγής.

```
private void addEventButton_Click(object sender, EventArgs e)
{
            addEventPanel.Visible = true;
            addEventButton.Visible = false;
}
```

Κώδικας 7 Εισαγωγή γεγονότος

Το παραπάνω απόσπασμα είναι υπεύθυνο για την εμφάνιση του Panel.



Εικόνες 29 Panel εισαγωγής γεγονότος

Αφού εισάγει τις πληροφορίες ο χρήστης το πρόγραμμα πρέπει να αναλάβει την αποθήκευση τους στην μνήμη και την ενημέρωση των κατάλληλων δομών δεδομένων. Για την αποθήκευση των γεγονότων επιλέχθηκε η κατασκευή μιας νέας κλάσης προκειμένου να ομαδοποιήσει τα γνωρίσματά τους.

```
//Class to store events for day
public class Event
{
        public DateTime Date { get; set; }
        public String Title { get; set; }
        public String Data { get; set; }
        public int TimeSched { get; set; }
        public Event()
        {

        }
        public Event(DateTime date, String title, String data,int timeSched)
        {
            Date = date;
            Title = title;
            Data = data;
            TimeSched = timeSched;
        }
}
```

Κώδικας 8 Κλάση γεγονός

Έπειτα χρειάστηκε μια δομή να αποθηκεύει όλα τα γεγονότα. Έτσι επιλέχθηκε η δομή της λίστα η οποία στην C# παρέχει έτοιμες μεθόδους για την διαχείριση των δεδομένων.

```
List<Event> eventsData = new List<Event>();
```

Κώδικας 9 Δήλωση της λίστας γεγονότων

Όταν ο χρήστης πατήσει το OK κουμπί που φαίνεται και στην εικόνα 13 καλείται ο παρακάτω listener.

```
private void eventButtonOK_Click(object sender, EventArgs e)
 {
     addEventPanel.Visible = false;
     addEventButton.Visible = true;
     DateTime date = dateTimePicker1.Value;
     String title = addEventTextBox1.Text;
     String data = addEventTextBox2.Text;
     int timeSched = addEventComboBox.SelectedIndex;

     Event di = new Event(date,title,data,timeSched);
     eventsData.Add(di);

     weekPanel_initComp();
     dayPanel_initComp();
     calendarPanel.Invalidate();
 }
```

Κώδικας 10 Action Listener OK button

3.2.3 Προβολή του Μήνα

Για την εμφάνιση του μήνα (το κεντρικό Tab Control Container) φάνηκε αρκετά χρήσιμη η διαδικασία Paint που παρέχει η C#. Πρόκειται για το πιο απαιτητικό στάδιο της εργασίας καθώς χρειάστηκαν αρκετές δοκιμές προκειμένου να εμφανιστούν στοιχισμένα και ομοιόμορφα τα στοιχεία του ημερολογίου. Σε αντίθεση με τα υπόλοιπα αντικείμενα της φόρμας το συγκεκριμένου είναι εξολοκλήρου φτιαγμένο με κώδικα, γεγονός που το κάνει πιο δύσκολο.

Έγινε χρήση της κλάσης RectangleF για την εμφάνιση των πλαισίων του ημερολογίου. Για την σωστή εμφάνιση των γραμματοσειρών μέσα στα πλαίσια υλοποιήθηκε η συνάρτηση FintFontSize η οποία υπολογίζει ποιο είναι το μεγαλύτερο μέγεθος γραμμάτων που χωράει ομαλά στα πλαίσια των ημερών , των μηνών και των γεγονότων. Η εν λόγω κλάση είναι η παρακάτω.

```
// Find the largest integer font size that will fit in the given space.
private int FindFontSize(Graphics gr, RectangleF rectf, string font_name, string
text)
{
            for (int font_size = 5; ; font_size++)
            {
                using (Font font = new Font(font_name, font_size))
                {
                    SizeF text_size = gr.MeasureString(text, font);
                    if ((text_size.Width > rectf.Width) ||
                        (text_size.Height > rectf.Height))
                        return font_size - 1;
                }
            }
}
```

Κώδικας 11 FindFontSize συνάρτηση

Στο πάνω αριστερό και πάνω δεξί άκρο του κεντρικού ημερολογίου έχουν τοποθετηθεί δυο κουμπιά προκειμένου να δίνεται η δυνατότητα στον χρήστη να μετακινείται στον προηγούμενο και στον επόμενο μήνα του έτους.

Εικόνες 30 Κουμπιά αλλαγής μήνα

Για την υλοποίηση αυτής της λειτουργίας είναι υπεύθυνοι δυο ακόμα Action Listeners. Στο συγκεκριμένο σημείο παρουσιάστηκε ένα πρόβλημα στις περιπτώσεις αλλαγής του τελευταίου και του πρώτου μήνα του έτους. Έτσι χρειάστηκαν ειδικοί έλεγχοι για την ομαλή λειτουργία της εν λόγω ενέργειας.

```
private void nextMonth_Click(object sender, EventArgs e)
{
            int newM = FirstOfMonth.Month + 1;
            int newY = FirstOfMonth.Year;
            if (newM == 13)
            {
                newM = 1;
                newY++;
            }

            DateTime first_of_month = new DateTime(newY, newM,1);

            FirstOfMonth = first_of_month;
            calendarPanel.Invalidate();
}

private void prevMonth_Click(object sender, EventArgs e)
{
            int newM = FirstOfMonth.Month - 1;
            int newY = FirstOfMonth.Year;
            if (newM == 0)
            {
                newM = 12;
                newY--;
            }

            DateTime first_of_month = new DateTime(newY, newM, 1);

            FirstOfMonth = first_of_month;
            calendarPanel.Invalidate();
}
```

Κώδικας 12 Αλλαγή μήνα

3.2.4 Προβολή εβδομάδας

Το κεντρικό Container περιέχει δυο καρτέλες. Η πρώτη είναι η προβολή του μήνα που περιεγράφηκε στην προηγούμενη ενότητα και η δεύτερη είναι η προβολή της εβδομάδας. Όπως και τα υπόλοιπα στοιχεία της εφαρμογής ομαδοποιούμε τα επιμέρους αντικείμενα (όπως labels, text fields κ.α.) με τη χρήση ενός Panel.

Για την εμφάνιση της εβδομάδας επιλέχθηκε το DataGridView προκειμένου να αντιστοιχίζεται η κάθε ώρα της ημέρας με τις μέρες της συγκεκριμένης εβδομάδας. Για την εισαγωγή των δεδομένων στο DattGridView είναι υπεύθυνος ο παρακάτω κώδικας.

```
//prepare week panel with current values to labels
private void weekPanel_initComp()
{
            //get number of current week
            int weekNum;
            DateTime d = monthCalendar1.SelectionRange.Start;
            weekNum = WeekNumber(d);

            //Add text to labels
            weeklabel2.Text = "Week ";
            weeklabel2.Text += weekNum;
            d = d.StartOfWeek(DayOfWeek.Sunday);
            weeklabel1.Text = d.ToString("dd.MM.yyy");
            d = d.AddDays(6);
            weeklabel1.Text += " - ";
            weeklabel1.Text += d.ToString("dd.MM.yyy");

            //Delete prev data
            weekGridView.Rows.Clear();
            //Add rows for time schedule to week view and prepare week panel
            weekGridView.Rows.Add("8.00 - 10.00");
            weekGridView.Rows.Add("10.00 - 12.00");
            weekGridView.Rows.Add("12.00 - 14.00");
            weekGridView.Rows.Add("14.00 - 16.00");
            weekGridView.Rows.Add("16.00 - 18.00");
            weekGridView.Rows.Add("18.00 - 20.00");
            weekGridView.Rows.Add("20.00 - 22.00");
            weekGridView.Rows.Add("22.00 - 24.00");

            //Display Data for the current week
            d = monthCalendar1.SelectionRange.Start;
            d = d.StartOfWeek(DayOfWeek.Sunday);
            foreach (Event di in eventsData)
            {
                if(di.Date.Date >= d.Date && di.Date.Date <= d.AddDays(6).Date)
                {
                    if (!di.Title.Equals(""))
                    {
                        weekGridView[(int)di.Date.DayOfWeek + 1,
di.TimeSched].Value = di.Title;
                    }
                }
            }
 }
```

Κώδικας 13 Week Panel Init

Η συνάρτηση weekNumber() που καλείται στην τρίτη γραμμή του κώδικα υπολογίζει τον αριθμό της συγκεκριμένης εβδομάδας βασισμένη στην επιλεγμένη ημερομηνία από το Month Calendar. Έπειτα δίνονται οι κατάλληλες ονομασίες στις ετικέτες του Panel.

Το DataGridView αποτελείται από γραμμές και στήλες. Οι γραμμές αντιπροσωπεύουν τις ώρες τις ημέρας (όπου για διευκόλυνση του χρήστη διατίθενται οκτώ δίωρα) και οι στήλες τις ημέρες της εβδομάδας. Στο περιεχόμενο του Grid εισάγονται τα δεδομένα που έχουν αποθηκευτεί στη λίστα των γεγονότων που αναφέρθηκε πιο πάνω. Αξίζει εδώ να σημειώσουμε την ευελιξία που προσφέρει η C# (σε σύγκριση με γλώσσες προγραμματισμού όπως η C) για την διάσχιση των δομών δεδομένων που διαθέτει.

3.2.5 Προβολή ημέρας

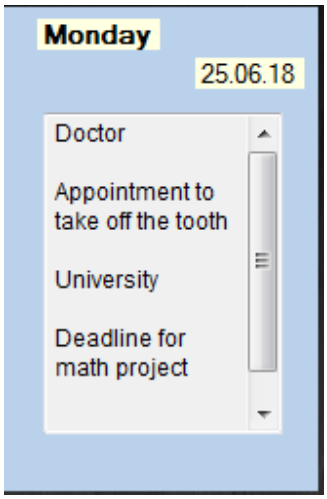
Το πρόγραμμα δίνει τη δυνατότητα στον χρήστη να ενημερώνεται για τα γεγονότα που έχει αποθηκεύσει σε μια συγκεκριμένη ημέρα. Για την συγκεκριμένη λειτουργία χρειάζεται ένα ακόμη Panel για τον ρόλο του Container όπου θα τοποθετηθούν οι ετικέτες και τα κείμενα που απαιτούνται.

Η συνάρτηση dayPanel_initComp() περιέχει τον κώδικα που εμφανίζει την ημερομηνία της ημέρας, τους τίτλους των γεγονότων και την περιγραφή τους.

```
private void dayPanel_initComp()
{
            DateTime d = monthCalendar1.SelectionRange.Start;
            int found = 0;
            dayInfoLabel.Text = d.DayOfWeek.ToString();
            dayInfoLabel2.Text = d.ToString("dd.MM.yy");
            dayDetailsTextBox.Text = "";
            foreach (Event ev in eventsData)
            {
                if(ev.Date.Date == d.Date)
                {
                    dayDetailsTextBox.Text += ev.Title;
                    dayDetailsTextBox.Text += Environment.NewLine;
                    dayDetailsTextBox.Text += Environment.NewLine;
                    dayDetailsTextBox.Text += ev.Data;
                    dayDetailsTextBox.Text += Environment.NewLine;
                    dayDetailsTextBox.Text += Environment.NewLine;
                    found = 1;
                }
            }
            if(found == 0)
            {
                dayDetailsTextBox.Text = "Nothing" + Environment.NewLine + "To " +
Environment.NewLine + "Do!!";
            }
}
```

Κώδικας 14 Day Panel Init

Ο παραπάνω κώδικας αρχικά λαμβάνει την τιμή που έχει επιλεχθεί από το Month Calendar. Έπειτα εμφανίζει τις ετικέτες για την ημέρα και την ημερομηνία. Εδώ παρατηρούμε την εύκολη διαχείριση των συμβολοσειρών στην C#. Το dayDetailsTextBox.text είναι το κείμενο που περιέχει όλους τους τίτλους και τις περιγραφές των γεγονότων που έχουν αποθηκευτεί για την συγκεκριμένη ημέρα. Μέσα στη εντολή foreach που ακολουθεί γίνεται διάσχιση της λίστα των γεγονότων και κάθε φορά που εντοπίζεται ένα αποθηκευμένο γεγονός στη συγκεκριμένη ημερομηνία εισάγετε στο κείμενο του TextBox ο τίτλος, δυο κενές γραμμές και τα δεδομένα που το περιγράφουν. Με την ακέραια μεταβλητή found γίνεται ο έλεγχος αν βρέθηκε ή όχι κάποιο γεγονός ώστε να εμφανιστεί το “Nothing To Do!”.



Εικόνες 31 Ημέρα με γεγονότα

4 Συμπεράσματα και επικείμενη βελτίωση

Κατά την περάτωση της πτυχιακής έγινε σαφής ο τρόπος που διαχειρίζονται οι προγραμματιστές τη γλώσσα προγραμματισμού C# και το Visual Studio. Η τριβή με μια ολοκληρωμένη αντικειμενοστραφής γλώσσα προγραμματισμού αποδίδει δεξιότητες που στη σύγχρονη επιστήμη της τεχνολογίας μοιάζουν απαραίτητες.

Η ενασχόληση με τις τεράστιες βιβλιοθήκες που παρέχει η C# και η χρήση τους στο περιβάλλον του Visual Studio είναι μια καλή γνωριμία με τις τεχνικές υλοποίησης εφαρμογών. Ιδιαίτερο ενδιαφέρον είχε ο συνδυασμός του “drag and drop” και της συγγραφής κώδικα.

Η εφαρμογή καλύπτει τις βασικές ανάγκες του χρήστη ενός ηλεκτρονικού Calendar ωστόσο επιδέχεται αρκετές προσθήκες και βελτιώσεις. Μερικές από τις αναβαθμίσεις που μπορούν να εκτελεστούν μελλοντικά είναι η εισαγωγή λίστας επαφών, η προβολή θρησκευτικών και εθνικών εορτών στο ημερολόγιο, η δυνατότητα κοινής χρήσης των δεδομένων της εφαρμογής με φίλους και συνεργάτες στο διαδίκτυο.

5 Βιβλιογραφία

1. Διαδικτυακός προγραμματισμός C#
   Ιωάννης – Χρήστος Π. Παναγιωτόπουλος 2009
2. https://el.wikipedia.org/wiki/C_Sharp
3. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
4. https://code.visualstudio.com/docs/languages/csharp
5. https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
6. https://www.tutorialspoint.com/csharp/
7. https://zapier.com/blog/best-calendar-apps/
8. https://en.wikipedia.org/wiki/Component-based_software_engineering
9. Patrick A.V. Hall, Martyn A. Ould, eds. (1996). Software Without Frontiers: A multi-platform, multi-cultural, multi-nation approach. With contributions and leadership by Ray Hudson, Costas Spyropoulos, Timo Honkela et al. Wiley.
10. C# Language Specification (4η έκδοση). Ecma International June 2006.
11. https://msdn.microsoft.com/en-us/library/zhe6k1tz.aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-3
12. https://www.lucidchart.com/pages/data-flow-diagram
13. https://www.thewindowsclub.com/add-national-holidays-calendar-app
14. https://signalvnoise.com/posts/1128-learning-from-bad-ui
15. https://www.sketchappsources.com/free-source/2346-calendar-ui-elements-sketch-freebie-resource.html
16. http://www.justscience.in/articles/hardware-software-borderline-cloud-computing/2018/01/22
17. https://honeycombsoft.com/Article/Detail/4