ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΟΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ (Πάτρα)

Τίτλος Εργασίας

Ανάπτυξη εφαρμογής κινητού τηλεφώνου για την απεικόνιση σε πραγματικό χρόνο δεδομένων από Web Service

Πτυχιακή Εργασία των

Αντωνόπουλος Γεώργιος - Α.Μ. 12265
Κωστόπουλος Παρασκευάς - Α.Μ. 12250

Επιβλέπων: Γιωτόπουλος Κωνσταντίνος

ΠΑΤΡΑ, ΙΟΥΝΙΟΣ 2015

ΕΥΧΑΡΙΣΤΙΕΣ

Θέλουμε να ευχαριστήσουμε τις οικογένειές μας για την στήριξη που μας παρείχαν καθ’ όλη τη διάρκεια των σπουδών μας καθώς και τους συμφοιτητές μας.

Επίσης ολοκληρώνοντας την πτυχιακή μας εργασία θα θέλαμε να ευχαριστήσουμε θερμά τους καθηγητές του τμήματος για την γνώση που μας μετέδωσαν και τον κύριο Γιωτόπουλο Κωνσταντίνο για την καθοδήγησή του στην εκπόνηση της εργασίας μας.

ΠΕΡΙΛΗΨΗ

Το διαδίκτυο και οι τεχνολογίες επικοινωνίας συμβαδίζουν αναπόφευκτα τα τελευταία χρόνια. Η ταχεία ανάπτυξή τους την τελευταία δεκαετία επέφερε αλλαγές και στην κινητή τηλεφωνία, όπου τα απλά κινητά τηλέφωνα εξελίχτηκαν σε έξυπνα κινητά τηλέφωνα (smartphones). Η δυναμική αύξηση των χρηστών smartphone έχει ως αποτέλεσμα την συνεχή δημιουργία όλο και περισσότερων νέων εφαρμογών και την ευρύτερη ποικιλία τους στις αντίστοιχες αγορές (smartphone application stores).

Με τη διάθεση λογισμικών για την δημιουργία smartphone εφαρμογών από τις κατασκευαστικές εταιρίες λειτουργικών συστημάτων, υπάρχει πλέον μεγάλη γκάμα εφαρμογών για ενημέρωση αλλά και ψυχαγωγία. Επιπλέον η πρόσβαση σε δεδομένα υψηλής ταχύτητας μέσω των τεχνολογιών Wi-Fi και κινητής πρόσβασης στο internet (3G, 4G - LTE) συνέβαλε στο να εξελιχθούν οι μέθοδοι επικοινωνίας μεταξύ εφαρμογών και web τεχνολογιών.

Η παρούσα πτυχιακή εργασία έχει ως κύριο στόχο την ανάπτυξη μιας εφαρμογής κινητού τηλεφώνου για την απεικόνιση του καιρού σε πραγματικό χρόνο από web service, μέσω ασύρματης πρόσβασης στο internet. Η εφαρμογή που υλοποιήθηκε ονομάστηκε 4Seasons και δίνει την δυνατότητα στον χρήστη να λαμβάνει πληροφορίες σχετικά με τα καιρικά φαινόμενα που επικρατούν την δεδομένη στιγμή στην τοποθεσία που βρίσκεται αλλά και στις περιοχές της επιλογής του, ακόμα και κατά την διάρκεια ενός ταξιδιού. Συγκεκριμένα χρησιμοποιήθηκε η γλώσσα προγραμματισμού JavaScript, περιεχομένου HTML, μορφοποίησης CSS και είναι συμβατή με το λειτουργικό σύστημα Windows Phone 8.1 της Microsoft.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εφαρμογή έξυπνου κινητού τηλεφώνου

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: Έξυπνα κινητά τηλέφωνα, αγορά εφαρμογών, υπηρεσίες web, windows phone 8.1, JavaScript, HTML, CSS

ABSTRACT

The internet inevitably has kept pace with communication technologies in recent years. Their rapid development in the last decade has brought about changes in the mobile telephony, where simple mobile phones have evolved to smartphones. The dynamic increase of smartphone users has resulted in the continuous creation of increasingly new applications and a wider variety in the respective markets (smartphone application stores).

With the disposal of software for the creation of smartphone applications by operating systems developing companies, there is a great range of applications not only for information but also for entertainment. Furthermore the access to high-speed data through Wi-Fi and mobile internet technologies has contributed to the evolution of communication methods between applications and web technologies.

The present dissertation mainly aims at the development of a smartphone application for the real time depiction of weather from web service through wireless internet access. The application implemented is called 4Seasons and provides the user with the potential of receiving information in relation to weather conditions prevailing the given moment at their current or favorite location and even during a journey. Specifically, JavaScript programming language, HTML content language and CSS style sheet language were used and the application is compatible with Microsoft Windows Phone 8.1 operating system.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Smartphone application

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: Smartphones, application stores, web services, windows phone 8.1, JavaScript, HTML, CSS

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΓΡΑΦΗΜΑΤΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Διανύοντας τον 21ο αιώνα, η γρήγορη τεχνολογική πρόοδος και οι συνεχώς αυξανόμενοι ρυθμοί ζωής των ανθρώπων χαρακτηρίζουν την κοινωνία στην οποία ζούμε. Η ανάγκη των ανθρώπων για άμεση επικοινωνία, γρήγορη ενημέρωση και ψυχαγωγία τους οδηγεί, παρά το φαινόμενο της παγκόσμιας οικονομικής κρίσης, να ακολουθούν πιστά την τεχνολογική πρόοδο αγοράζοντας συστήματα τελευταίας τεχνολογίας. Οι κλάδοι της πληροφορικής και των τηλεπικοινωνιών εξελίσσονται συνεχώς με την πάροδο του χρόνου τα τελευταία χρόνια διαγράφοντας έτσι μια ανοδική πορεία. Η ενσωμάτωση των τεχνολογιών της πληροφορικής στο χώρο των τηλεπικοινωνιών και ειδικότερα στις συσκευές κινητής τηλεφωνίας είχε ως αποτέλεσμα τα κινητά τηλέφωνα να αποκτήσουν νέες δυνατότητες προκειμένου να καλύψουν όσο το δυνατόν καλύτερα τις ανάγκες των χρηστών. Η ανάγκη για ενημέρωση μέσω των έξυπνων κινητών τηλεφώνων από υπηρεσίες που μας διαθέτει το διαδίκτυο γίνεται όλο και πιο επιτακτική. Συγκεκριμένα καλούμαστε να λύσουμε το πρόβλημα απεικόνισης δεδομένων σε μια εφαρμογή κινητού τηλεφώνου μέσω ενός Web Service σε πραγματικό χρόνο.

Πλέον οι συσκευές κινητής τηλεφωνίας ενσωματώνουν τεχνολογίες κινητού διαδικτύου, GPS, φωτογραφική μηχανή και άλλα πολλά. Η εξέλιξη τους σε έξυπνα κινητά τηλέφωνα με την ευρεία γκάμα υπηρεσιών που παρέχουν επηρέασε την παγκόσμια αγορά και εκτιμάται πως το 2014 πουλήθηκαν πάνω από 1 δισεκατομμύριο smartphones παγκοσμίως. Ως αποτέλεσμα αυτής της τεχνολογικής προόδου δημιουργήθηκε η ανάγκη των ηλεκτρονικών καταστημάτων εφαρμογών (App Stores). Με την αγορά ενός smartphone τα αντίστοιχα καταστήματα εφαρμογών ανά λειτουργικό σύστημα είναι διαθέσιμα σε όλους. Αυτό έχει ως αποτέλεσμα να δημιουργηθούν εφαρμογές με πολύ μεγάλες δυνατότητες και οι χρήστες να ικανοποιούν το μεγαλύτερο κομμάτι των αναγκών τους μέσα από αυτές. Όσο για τις εφαρμογές ενημέρωσης, απαιτούν τη σύνδεση στο διαδίκτυο και σχεδόν πάντα την επικοινωνία με ένα Web Service. Οι εφαρμογές που συνεργάζονται με Web Service έχουν βελτιωθεί τα τελευταία χρόνια και έχουν σημαντική επέκταση σε πολλούς τομείς μεταφέροντας κάθε είδους ενημέρωσης στην οθόνη ενός smartphone όπως την ενημέρωση καιρικών φαινομένων, χρηματιστηριακών εξελίξεων, ημερήσιας φθηνότερης τιμής αεροπορικών εισιτηρίων μέχρι και τον ακριβή χρόνο που θέλει το λεωφορείο για να φτάσει στην στάση.

Η εφαρμογή που δημιουργήσαμε αφορά το λειτουργικό σύστημα Windows Phone 8.1 και το αντίστοιχο κατάστημα εφαρμογών Windows Store. Η υλοποίηση της στη τελευταία έκδοση του Windows Phone OS θα επιφέρει γνώση για τη νέα αρχιτεκτονική του λειτουργικού, τα είδη εφαρμογών που μπορούν να αναπτυχθούν σε αυτό καθώς και το νέο περιβάλλον εκτέλεσης (Windows Runtime). Επιπλέον, θα μπορεί να είναι διαθέσιμη μόνο στο Windows Store, ένα ανερχόμενο κατάστημα που φιλοξενεί εφαρμογές για όλες τις συσκευές με αυτό το λειτουργικό. Η δυνατότητα ανάπτυξης μιας Windows Store εφαρμογής με πολλές γλώσσες προγραμματισμού και η επιλογή λύσης του προβλήματος με τη γλώσσα προγραμματισμού JavaScript θα μας γνωστοποιήσει αντικείμενα της νέας βιβλιοθήκης WinJS σε σχέση με το νέο περιβάλλον εκτέλεσης WinRT.

Το σύνολο των κεφαλαίων της πτυχιακής εργασίας είναι τέσσερα. Στο πρώτο κεφάλαιο γίνεται η βιβλιογραφική ανασκόπηση της εξέλιξης του κινητού τηλεφώνου έως και σήμερα και η γνωστοποίηση των τεχνολογιών σχετικά με τη λύση του προβλήματος. Στο δεύτερο κεφάλαιο γίνεται μια προσέγγιση στο λειτουργικό Windows Phone, στην αρχιτεκτονική του, στα είδη εφαρμογών που μπορούν να αναπτυχθούν και στα εργαλεία ανάπτυξης της εφαρμογής. Στο τρίτο κεφάλαιο καταγράφεται η ανάλυση, η σχεδίαση, η υλοποίηση της εφαρμογής και παραδείγματα χρήσης της. Τέλος στο τέταρτο κεφάλαιο αναφέρονται τα συμπεράσματα από την εκπόνηση της πτυχιακής και μελλοντικές επεκτάσεις της εφαρμογής.

ΚΕΦΑΛΑΙΟ 1. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

1.1. Έξυπνες κινητές συσκευές (Smartphones)

1.1.1. Η ιστορία των smartphones

Τα smartphones είναι κινητά τηλέφωνα βασισμένα σε λειτουργικό σύστημα κινητής τηλεφωνίας με περισσότερη προηγμένη υπολογιστική ικανότητα και συνδεσιμότητα σε σχέση με ένα απλό κινητό τηλέφωνο. Τα πρώτα smartphones συνδύαζαν τις λειτουργίες ενός προσωπικού ψηφιακού βοηθού και ενός κινητού τηλεφώνου. Με το πέρασμα των χρόνων προστέθηκαν και άλλες λειτουργίες όπως για παράδειγμα μονάδες πλοήγησης GPS, οθόνες αφής υψηλής ανάλυσης και φυλλομετρητές ιστοσελίδων. Τα τελευταία χρόνια η ταχεία ανάπτυξη στην αγορά των εφαρμογών (application stores) για κινητά, έγινε οδηγός για την ευρεία υιοθέτηση των smartphones στην παγκόσμια αγορά και στο εμπόριο κινητών τηλεφώνων(3)(4).

Η περιπέτεια της κινητής τηλεφωνίας ξεκίνησε αμέσως μετά τον Β' Παγκόσμιο Πόλεμο , με τις πρώτες προσπάθειες των Σουηδών, Φινλανδών και Αμερικάνων. Όμως ως ληξιαρχική πράξη γέννησής της θεωρείται η 3η Απριλίου 1973. Σχεδόν δέκα χρόνια μετά από την επίδειξη της πρωτότυπης συσκευής στην οποία βασίστηκε, κυκλοφορεί στην αγορά το πρώτο κινητό τηλέφωνο Motorola DynaTAC σχεδιασμένο από τον Martin Cooper. Η Motorola κατά τη διάρκεια της δεκαετής μεταβατικής περιόδου δαπάνησε περίπου 100 εκατομμύρια δολάρια για την ανάπτυξη της πρώτης κινητής συσκευής. Το 1983 το κοινό ήταν έτοιμο να υποδεχτεί αυτήν τη νέα τεχνολογία και παρά τα 4.000 δολάρια της αρχικής του τιμής , το DynaTAC 8000X έτυχε θερμότατης υποδοχής.

Εικόνα 1-1: Πρώτο κινητό τηλέφωνο Motorola Dynatac 8000x - Martin Cooper(5).

Με τα σημερινά δεδομένα, οι δυνατότητες του φαντάζουν πρωτόγονες. Η μπαταρία του διαρκούσε για μόλις 30 λεπτά ομιλίας, ενώ απαιτούσε φόρτιση 10 ωρών. Τέλος, διέθετε οθόνη ερυθρών LED και μνήμη για 30 αριθμούς τηλεφώνων. Το Motorola DynaTac 8000X έμελλε να είναι η ναυαρχίδα των κινητών τηλεφώνων πρώτης γενιάς (1G). Το 1993 κυκλοφόρησε το «Simon Personal Communicator», όπως ήταν το πλήρης όνομα της συσκευής είναι ιστορικά το πρώτο smartphone της αγοράς. Αυτή ήταν και η πρώτη προσπάθεια για να χωρέσουν σε μια μικρή κινητή συσκευή οι λειτουργίες ενός κινητού τηλεφώνου, υπολογιστή και Fax. Επίσης, πρέπει να σημειωθεί πως ήταν η πρώτη συσκευή που έγινε απόπειρα για χρήση εφαρμογών σε smartphone (apps) (5).

Στα μέσα της δεκαετίας του ’90 περάσαμε στα κινητά τηλέφωνα δεύτερης γενιάς (2G). Τότε άρχισε η απογείωση των κινητών τηλεφώνων, με την ψηφιοποίηση δικτύων (GSM) και συσκευών που παρείχαν και άλλες ευκολίες, όπως την αποστολή σύντομων γραπτών μηνυμάτων (SMS) και τη λήψη φωτογραφιών (MMS). Τα κινητά τηλέφωνα έγιναν μικρότερα και ελαφρύτερα κατά 100-200 γραμμάρια με αποτέλεσμα να χωράνε στην παλάμη και με δυσκολία στην τσέπη του χρήστη τους.

Εικόνα 1-2: Κινητό τηλέφωνο 2ης γενιάς - Nokia 3310(48).

Στις αρχές 21ου αιώνα ήλθαν τα κινητά τρίτης γενιάς (3G), με κύριο χαρακτηριστικό τις απεριόριστες δυνατότητες των πολυμέσων. Έτσι άρχισε η μαζική παραγωγή κινητών συσκευών και η μεταβατική περίοδος των απλών κινητών τηλεφώνων σε μια νέα εποχή, την εποχή των smartphones. Η διάκριση μεταξύ των smartphones και των απλών κινητών τηλεφώνων μπορεί να είναι ασαφής και δεν υπάρχει επίσημος ορισμός για το ποιες είναι οι μεταξύ τους διαφορές. Μία από τις πιο σημαντικές διαφορές είναι ότι οι προηγμένες διεπαφές προγραμματισμού εφαρμογών (APIs) στα smartphones, σχετικά με την λειτουργία τρίτων εφαρμογών, μπορούν να

επιτρέψουν την καλύτερη ενσωμάτωση στο λειτουργικό σύστημα και στο hardware του τηλεφώνου απ' ότι συμβαίνει συνήθως στα απλά κινητά τηλέφωνα. Η πρώτη συσκευή που έμελλε να ταράξει τα νερά στο χώρο της κινητής τηλεφωνίας ήταν το iPhone 3G το οποίο παρουσιάστηκε το 2007 από τον τότε πρόεδρο και διευθύνων σύμβουλο της Apple, Steve Jobs. Το iPhone 3G ήταν το πρώτο smartphone που σηματοδοτούσε την έναρξη της εποχής των έξυπνων κινητών τηλεφώνων. Είχε μεταξύ άλλων οθόνη αφής 3.5 ιντσών, Wi-Fi, Bluetooth και κάμερα 2 Megapixel(6).

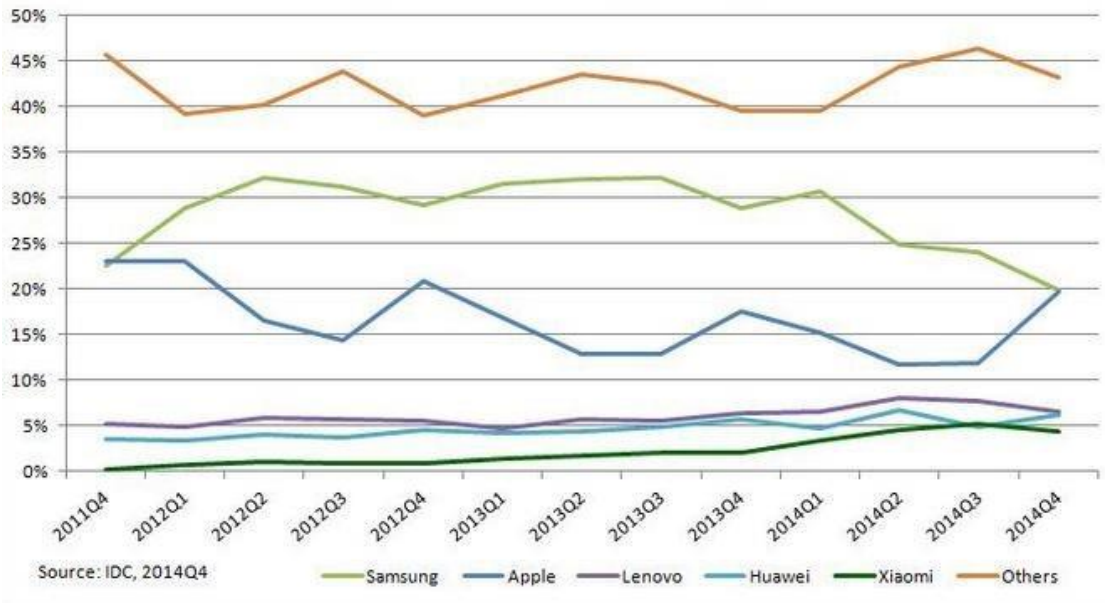
Εικόνα 1-3: Κινητό τηλέφωνο 3ης γενιάς - Apple iPhone 3G(49).

Ως επερχόμενη γενιά κινητών τηλεφώνων θεωρείται η τέταρτη, έχοντας την πιο εξελιγμένη μέχρι τώρα τεχνολογία. Ο συνδυασμός τους με την πιο γρήγορη mobile πρόσβαση στο internet (δίκτυα 4ης γενιάς) δίνει μια δυναμική στις εφαρμογές υπηρεσιών (μετεωρολογική ενημέρωση, GPS), περιεχομένου, ψυχαγωγίας (ειδήσεις, ραδιόφωνο, παιχνίδια) αλλά και πρόσβασης στις ιστοσελίδες κοινωνικής δικτύωσης (π.χ. Facebook, Twitter) όπου χαρακτηρίζονται από την αμεσότητα και την ταχύτητά τους. Στην ουσία οι δυνατότητες των έξυπνων κινητών τηλεφώνων τέταρτης γενιάς, που ξεκίνησε το 2010 έως και σήμερα, δεν διαφέρουν από αυτές ενός υπολογιστή.

Εικόνα 1-4: Έξυπνα κινητά τηλέφωνα του 2015(50).

1.1.2. Η αγορά των smartphones

Η αυξημένη ζήτηση από τους τελικούς χρήστες και οι απεριόριστες επιλογές μοντέλων smartphones που προσφέρονται προς πώληση οδήγησαν σε ένα νέο ρεκόρ όγκου παραγωγής για το έτος 2014. Σύμφωνα με στοιχεία από την International Data Corporation (IDC) η παγκόσμια αγορά των smartphones αυξήθηκε κατά 28.2% σε ετήσια βάση κατά το τέταρτο τρίμηνο του 2014 (2014Q4) με αριθμό πωλήσεων περίπου 377.5 εκατομμύρια μονάδες. Η Apple κατάφερε να παραδώσει 74.5 εκατομμύρια συσκευές το τέταρτο τρίμηνο του 2014 με μία ετήσια αύξηση της τάξεως του 46% κλείνοντας έτσι το κενό μεταξύ της Samsung. Οι συνολικές παγκόσμιες πωλήσεις του έτους ανήλθαν σε 1.3 δισεκατομμύρια συσκευές έξυπνων κινητών τηλεφώνων, δηλαδή 27.7% περισσότερο από το 2013.



Γράφημα 1-1: Παγκόσμια αγορά smartphones ανά κατασκευαστή την περίοδο 2011-2014 (σε %) (7).

Η **Samsung** διατήρησε την πρώτη θέση στην παγκόσμια αγορά των smartphone κατά το τέταρτο τρίμηνο με 0.2% διαφορά από την Apple. Ο ισχυρός ανταγωνισμός από την Apple καθώς και η επιβράδυνση της ζήτησης είχε αρνητικό αποτέλεσμα στα τρία τελευταία τρίμηνα του έτους 2014. Με σκοπό να ανακτήσει το μερίδιο της στη κορυφή, η Samsung θα πρέπει να διερευνήσει καινοτομίες, να υιοθετήσει χαμηλότερα περιθώρια κέρδους και να αναδιοργανώσει την στρατηγική της.

Η **Apple** έφτασε σε ένα νέο ρεκόρ για το τρίτο τρίμηνο του 2014 ξεπερνώντας τη Samsung, εκπλήσσοντας έτσι την παγκόσμια αγορά έξυπνων κινητών τηλεφώνων. Η μεγάλη προβολή των συσκευών της Apple είχε ως αποτέλεσμα την αύξηση 44% των

πωλήσεων των iPhone στην Αμερική και των διπλασιασμό των πωλήσεων στις BRIC (Brazil, Russia, India, China). Αυτό που απομένει είναι να δούμε για πόσο θα μπορέσει η εταιρεία να διατηρήσει την ανάπτυξη της για το υπόλοιπο του 2015.

Η **Lenovo** με μεγάλη διαφορά από τους πρωτοπόρους της αγοράς (Samsung, Apple), είναι τρίτη στην παγκόσμια αγορά smartphone χάρη στην ολοκλήρωση της εξαγοράς της Motorola. Παρά το γεγονός ότι είχε ελαφρώς λιγότερες πωλήσεις το τέταρτο τρίμηνο του 2014 σε σχέση με το τρίτο συνέχισε να κυριαρχεί στις πωλήσεις έξυπνων κινητών τηλεφώνων με κόστος κάτω των 150$ στην Κίνα με συσκευές όπως το Golden Warriors S8 και το Vibe Z2 pro. Η Lenovo ανακοίνωσε πρόσφατα ότι θα φέρει το εμπορικό σήμα της Motorola πίσω στην Κίνα, αρχής γενομένης με το Moto X τον Ιανουάριο του 2015.

Η **Huawei** αναγεννημένη στη τέταρτη θέση, δίνοντας έμφαση στα μεσαίου κόστους smartphone (P Series και Mate Series) είδε να συνεχίζεται η επιτυχία της. Η Huawei απέδωσε την επιτυχία του 2014 στην βελτιωμένη εικόνα της εταιρείας και στη συνολική εμπειρία του πελάτη μέσα από τις συσκευές της, τις οποίες θα προσπαθήσει να βελτιώσει μέσα στο 2015.

Η **Xiaomi** τέλος, είναι η πέμπτη δύναμη στο χώρο των έξυπνων κινητών τηλεφώνων για το τέταρτο τρίμηνο του 2014 παρότι ο όγκος πωλήσεων της μειώθηκε ελαφρώς από το τρίτο τρίμηνο του ίδιου έτους. Η Xiaomi κατέχει το υψηλότερο ποσοστό ετήσιας ανάπτυξης (περίπου 178.3%) ανάμεσα στους κορυφαίους κατασκευαστές, χάρη στην αυξημένη ζήτηση της εγχώριας Κινέζικης αγοράς. Με μια σταθερή απελευθέρωση του χαμηλού κόστους αλλά και με άκρως ανταγωνιστικά προϊόντα, όπως το Mi4 LTE, η πέμπτη θέση της κρατιέται από μια κλωστή(7).

Period	Samsung	Apple	Lenovo*	Huawei	Xiaomi	Others
Q4 2014	19.9%	19.7%	6.5%	6.3%	4.4%	45.7%
Q4 2013	28.9%	17.5%	6.4%	5.7%	2.0%	39.5%
Q4 2012	29.1%	20.9%	5.5%	4.6%	0.9%	39.0%
Q4 2011	22.5%	23.0%	5.1%	3.5%	0.1%	45.7%

Πίνακας 1-1: Παγκόσμια αγορά smartphones ανά κατασκευαστή την περίοδο 2011-2014 (σε %) (7).

1.1.3. Λειτουργικά συστήματα των smartphones

Λειτουργικό σύστημα (Operating System ή OS) ονομάζεται στην επιστήμη της πληροφορικής το λογισμικό του υπολογιστή που είναι υπεύθυνο για τη διαχείριση και τον συντονισμό των εργασιών, καθώς και την κατανομή των διαθέσιμων πόρων. Παρέχει ένα θεμέλιο δηλαδή ένα μεσολαβητικό επίπεδο λογικής διασύνδεσης μεταξύ λογισμικού και υλικού, διαμέσου του οποίου οι εφαρμογές αντιλαμβάνονται εμμέσως τον υπολογιστή. Μια από τις κεντρικές αρμοδιότητές του είναι η διαχείριση του υλικού, απαλλάσσοντας έτσι το λογισμικό του χρήστη από τον άμεσο και επίπονο χειρισμό του υπολογιστή και καθιστώντας ευκολότερο τον προγραμματισμό τους. Σχεδόν όλοι οι υπολογιστές (παλάμης, επιτραπέζιοι, υπερυπολογιστές, ακόμη και παιχνιδομηχανές) χρησιμοποιούν έναν τύπο λειτουργικού συστήματος. Ορισμένα παλαιότερα μοντέλα ωστόσο βασίζονται σε ένα ενσωματωμένο λειτουργικό σύστημα, το οποίο περιέχεται σε έναν οπτικό δίσκο ή άλλες συσκευές αποθήκευσης δεδομένων.

Ως λειτουργικό σύστημα έξυπνων κινητών τηλεφώνων χαρακτηρίζεται μία συλλογή βασικών προγραμμάτων, η οποία ελέγχει τη λειτουργία του smartphone συνολικά και χρησιμοποιείται ως υπόβαθρο για την εκτέλεση όλων των υπόλοιπων προγραμμάτων, τη διαχείριση των περιφερειακών συσκευών και την εξασφάλιση της επικοινωνίας μεταξύ χρήστη και συσκευής. Στην πράξη πρόκειται για ένα επίπεδο λογισμικού που μεσολαβεί μεταξύ υλικού και εκτελούμενων προγραμμάτων σε κάθε κινητή συσκευή. Αποτελείται από ένα σύνολο μηχανισμών μέσω των οποίων επιτυγχάνεται αυτόματη διαχείριση και ελεγχόμενη κατανομή στις εκτελούμενες εφαρμογές, έτσι ώστε οι τελευταίες να είναι σε θέση να προσπελάσουν εύκολα τους πόρους και τις συσκευές του συστήματος χωρίς να χρειάζεται να γνωρίζουν με ακρίβεια τη δομή του υποκείμενου υλικού, αλλά και ώστε πολλαπλές εφαρμογές να μπορούν να εκτελούνται ταυτόχρονα χωρίς να έρχονται σε διένεξη μεταξύ τους ή με τον υπολογιστή.

Οι πρώιμοι υπολογιστές στερούνταν λειτουργικού συστήματος, ένας άνθρωπος «χειριστής» (operator) φόρτωνε τα προγράμματα στη μνήμη του υπολογιστή και φρόντιζε για την εκτέλεσή τους, Το λειτουργικό σύστημα ήρθε να αντικαταστήσει τις ανθρώπινες αυτές ενέργειες. Με το μεσολαβητικό επίπεδο διασύνδεσης που προσφέρει, οι εφαρμογές μπορούν να αξιοποιούν εύκολα τη μνήμη, τον επεξεργαστή, το σύστημα αρχείων και τις περιφερειακές συσκευές. Δηλαδή δημιουργεί ένα απλουστευμένο εικονικό περιβάλλον μέσα στο οποίο εκτελούνται οι εφαρμογές προκειμένου να φέρουν εις πέρας τις εργασίες που ζητά ο χρήστης(8).

Στην συνέχεια θα γίνει μια συνοπτική περιγραφή των πιο διαδεδομένων λειτουργικών συστημάτων για smartphone:

Το **Android** είναι σχεδιασμένο για να προσφέρει τη μέγιστη παραμετροποίηση στα κινητά τηλέφωνα, ανάλογα με τις επιθυμίες των χρηστών. Προσφέρει την δυνατότητα χρήσης εξελιγμένων υπηρεσιών του διαδικτύου, όπως ανταλλαγή μηνυμάτων ηλεκτρονικού ταχυδρομείου, πρόσβαση σε σελίδες κοινωνικής δικτύωσης, πλοήγηση στο Internet και φυσικά ψυχαγωγία. Υπόσχεται πληθώρα εφαρμογών που θα βελτιώσουν την εμπειρία της αναπαραγωγής μουσικής, της διαχείρισης αρχείων, της λήψης φωτογραφιών και βίντεο αλλά και της επικοινωνίας μέσω SMS, MMS και φωνητικών κλήσεων. Βασίζεται στον πυρήνα του λειτουργικού συστήματος Linux και πρόκειται για μια σύμπραξη εταιρειών-κολοσσών στον τομέα της τεχνολογίας, όπως η Google, η Motorola, η HTC, η Samsung, η LG, η Qualcomm, η Sprint, η T-Mobile και άλλες, οι οποίες στις 5 Νοεμβρίου του 2007 έφεραν στο φως το νέο αυτό λειτουργικό σύστημα. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού, γεγονός που επιτρέπει σε πλήθος προγραμματιστών να σχεδιάσουν εφαρμογές για το συγκεκριμένο λειτουργικό σύστημα με τη χρήση της γλώσσας προγραμματισμού Java.

Το **iOS** (γνωστό και ως iPhone OS πριν από τον Ιούνιο 2010) είναι το λειτουργικό σύστημα της Apple. Αναπτύχθηκε για το iPhone και παρουσιάστηκε στις 9 Ιανουαρίου του 2007 και προέρχεται από το Mac OS που χρησιμοποιεί η Apple στους υπολογιστές της και συνεπώς είναι ένα λειτουργικό σύστημα τύπου Unix. Από τότε έχει επεκταθεί για να υποστηρίξει και άλλες συσκευές της Apple όπως το iPod touch, iPad και Apple TV. Η Apple δεν χορηγεί άδεια για εγκατάσταση του λειτουργικού σε τρίτους κατασκευαστές όμως αυτό δεν την καθιστά μικρό παίχτη στην αγορά. Το περιβάλλον του iOS είναι βασισμένο στον άνετο χειρισμό, στην απλότητα και αυτό που το χαρακτηρίζει είναι η άμεση ανταπόκρισή του στην διάθεση του χρήστη όπως και η αδιάκοπη λειτουργία του.

Τα **Windows Phone** είναι το λειτουργικό σύστημα που αναπτύχθηκε από την Microsoft και αποτελεί τον διάδοχο των Windows Mobile. Χρησιμοποιήθηκαν αρχικά στην Ευρώπη, την Σιγκαπούρη, την Αυστραλία και τη Νέα Ζηλανδία στις 21 Οκτωβρίου 2010, στις ΗΠΑ και τον Καναδά στις 8 Νοεμβρίου 2010, στο Μεξικό στις 24 Νοεμβρίου 2010 και στην Ασία μέσα στο τρέχον έτος. Η επίσημη αποκάλυψη έγινε στη Βαρκελώνη στις 15 Φεβρουαρίου 2010, ενώ έγινε διαθέσιμο για τους κατασκευαστές και προγραμματιστές στις 16 Σεπτεμβρίου του ίδιου έτους. Εδώ, για να

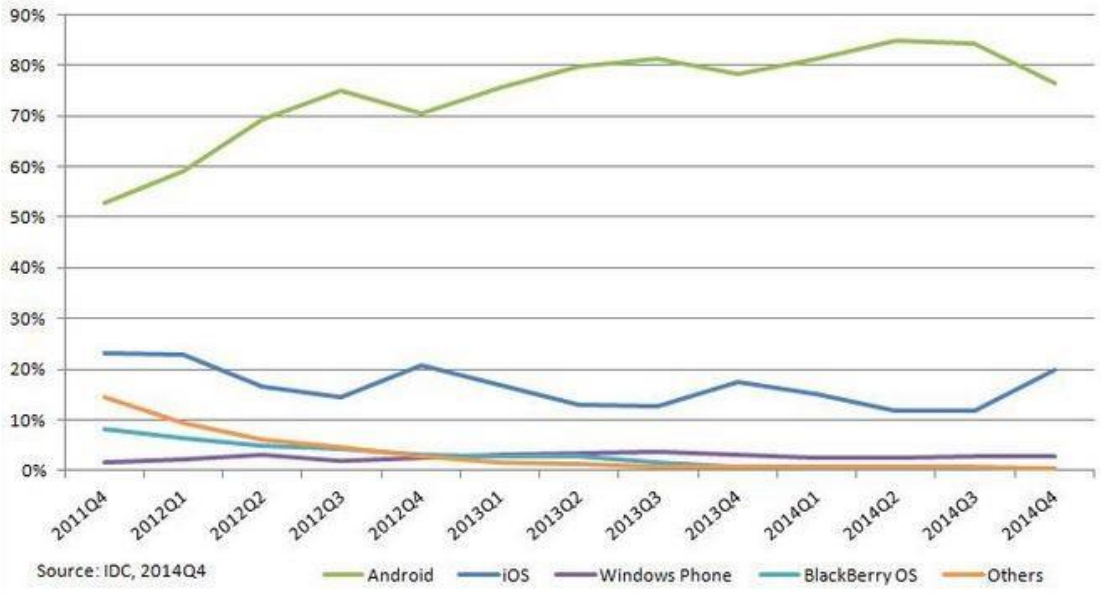
αποφευχθεί οποιαδήποτε παρανόηση, πρέπει να σημειωθεί πως τα Windows Phone είναι ο αντικαταστάτης των Windows Mobile, η τελευταία έκδοση των οποίων ήταν η 6.5. Με τα Windows Phone, η Microsoft προσφέρει ένα νέο περιβάλλον, με τη σχεδιαστική γλώσσα που η ίδια έχει αναπτύξει και την ονόμασε Metro. Η σχεδιαστική αυτή γλώσσα δημιουργήθηκε ως διεπαφή της Microsoft για το Zune και για τα Windows Media Center κατά την ενσωμάτωσή τους στο νέο λειτουργικό Windows Phone. Όπως συμβαίνει και με τα άλλα λειτουργικά, έτσι και στα Windows Phone, υπάρχει το αντίστοιχο application store όπου ο χρήστης κατεβάζει εφαρμογές τόσο από την Microsoft, όσο και από τρίτους κατασκευαστές. Τέλος, δεν θα πρέπει να ξεχνάμε την ανακοίνωση της Microsoft στις 11 Φεβρουαρίου 2011 στο Λονδίνο για την συνεργασία της με την Nokia όπου αφήνει πίσω της το λειτουργικό σύστημα Symbian και χρησιμοποιεί ως βασικό της λειτουργικό, για τα κινητά, το Windows Phone.

Το **Web OS** είναι ένα λειτουργικό σύστημα που βασίζεται στον πυρήνα του Linux. Αρχικά αναπτύχθηκε από την Palm, αλλά στη συνέχεια λόγω της εξαγοράς της Palm από την HP, άλλαξε χέρια και το WebOS. Το λειτουργικό αυτό είχε εισαχθεί από την Palm τον Ιανουάριο του 2009 ως διάδοχος του Palm OS και αναγνωρίστηκε από το αγοραστικό κοινό για την ενσωμάτωση της τεχνολογίας Web 2.0, την ανοικτή του αρχιτεκτονική, τις multitasking δυνατότητες και την ευκολία χρήσης του. Η πρώτη συσκευή που το χρησιμοποίησε ήταν το Palm Pre, το οποίο κυκλοφόρησε από την Sprint τον Ιούνιο του 2009. Επίσης, ήταν ένα από τα πρώτα λειτουργικά συστήματα που έκαναν ευρεία χρήση των over-the-air ενημερώσεων για όλες τις συσκευές που το χρησιμοποιούσαν. Τον Φεβρουάριο 2011, η HP ανακοίνωσε μια σειρά από νέες συσκευές που θα χρησιμοποιούν για πρώτη φορά το λειτουργικό σύστημα WebOS 2.2 συμπεριλαμβανομένου των smartphones HP Pro 3 και HP Veer.

Το **BlackBerry OS** είναι ένα λειτουργικό σύστημα, που αναπτύχθηκε από την Research In Motion για τα κινητά Blackberry. Παρέχει multitasking και υποστηρίζει τις εξειδικευμένες συσκευές εισόδου που έχουν υιοθετηθεί από την RIM και τη χρήση αυτών σε φορητές συσκευές της, όπως το trackball, το trackpad και η οθόνη αφής. Όπως στα περισσότερα λειτουργικά, έτσι και στο Blackberry OS, οι ενημερώσεις για το λειτουργικό σύστημα μπορούν επίσης να γίνουν με την τεχνολογία over-the-air. Επίσης, δίνεται η δυνατότητα σε τρίτους προγραμματιστές να γράψουν λογισμικό χρησιμοποιώντας τα διαθέσιμα BlackBerry API(9).

1.1.4. Η αγορά των λειτουργικών συστημάτων

Όταν είχαν πρωτοεμφανιστεί τα κινητά, τα τεχνικά χαρακτηριστικά τους ήταν αυτά που μας οδηγούσαν στην αγορά τους. Πλέον τα δεδομένα έχουν αλλάξει και μεγάλο ποσοστό στην τελική απόφαση του νέου μας κινητού παίζει το λειτουργικό σύστημα που φέρει. Σύμφωνα με τα τελευταία δεδομένα της International Data Corporation (IDC) οι πωλήσεις smartphone ανά λειτουργικό σύστημα είχαν την εξής εξέλιξη.



Γράφημα 1-2: Παγκόσμια αγορά smartphones ανά λειτουργικό σύστημα την περίοδο 2011-2014 (σε %) (10).

Το **Android** πέρασε το φράγμα του ενός δισεκατομμυρίου μονάδων το 2014, ένα σημαντικό ορόσημο από μόνο του , αλλά και επειδή ο συνολικός όγκος Android συσκευών του 2014 ξεπέρασε εκείνον του 2013. Η Samsung διατήρησε την ηγετική της θέση με μεγάλη διαφορά αποστέλλοντας πολύ περισσότερες συσκευές από τις υπόλοιπες εταιρείες μαζί. Την ίδια στιγμή, ο συνολικός όγκος πωλήσεων της Samsung για το υπόλοιπο του έτους (2014) παρέμεινε ο ίδιος, ενώ αντίστοιχα οι Ασιατικές εταιρείες συμπεριλαμβανομένου της Huawei, Lenovo, LG Electronics, Xiaomi και ZTE συνέβαλαν περισσότερο στην μεγαλύτερη ανάπτυξη της κονσόλας της Google.

Το **iOS** είδε το μερίδιο αγοράς του για το 2014 ελαφρώς μειωμένο, ακόμη και αν όγκος πωλήσεων έφθασε σε νέο ρεκόρ και αναπτύχθηκε σχεδόν με το ίδιο ρυθμό όπως και το σύνολο της αγοράς smartphone. Μεγάλο μέρος αυτού οφείλεται στο ισχυρό αίτημα για νέες και μεγαλύτερες iPhone συσκευές καθώς και στην απήχηση που είχαν στην αγορά.

Το **Windows Phone** λογισμικό της Microsoft είχε ετήσια αύξηση, περίπου 4.2%, μεταξύ των κορυφαίων λειτουργικών συστημάτων. Αφού ολοκλήρωσε την εξαγορά της Nokia, την άνοιξη του 2014, η Microsoft στηρίχθηκε κυρίως στη σειρά έξυπνων κινητών τηλεφώνων Lumia ώστε να διατηρήσει τη θέση της στην αγορά.

Το **Blackberry OS** ήταν το μοναδικό λογισμικό μεταξύ των κορυφαίων εταιρειών στα λειτουργικά συστήματα που σημείωσε ετήσια πτώση -69.8% από τα επίπεδα του 2013. Το 2014 σηματοδότησε τον εξ ορθολογισμό για την πλατφόρμα που βρισκόταν σε αδιέξοδο και μέχρι το τέλος του έτους η εταιρεία αποκάλυψε πολλαπλές βελτιώσεις στην πλατφόρμα της και την προσθήκη δύο ακόμα συσκευών, το BlackBerry Passport και το BlackBerry Classic όπου παρουσιάστηκαν από την εταιρεία(10).

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q4 2014	76.6%	19.7%	2.8%	0.4%	0.5%
Q4 2013	78.2%	17.5%	3.0%	0.6%	0.8%
Q4 2012	70.4%	20.9%	2.6%	3.2%	2.9%
Q4 2011	52.8%	23.0%	1.5%	8.1%	14.6%

Πίνακας 1-2: Παγκόσμια αγορά smartphones ανά λειτουργικό σύστημα την περίοδο 2011-2014 (σε %) (10).

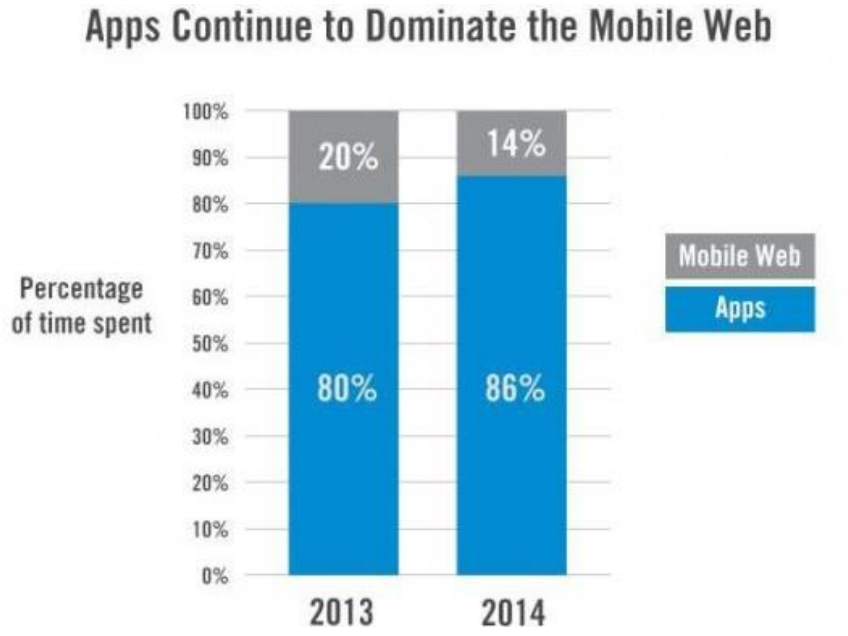
1.1.5. Καταστήματα εφαρμογών

Ένα ηλεκτρονικό κατάστημα εφαρμογών (app store) είναι ένα είδος ψηφιακής πλατφόρμας διανομής εφαρμογών για κινητά τηλέφωνα. Οι εφαρμογές παρέχουν ένα συγκεκριμένο σύνολο που, εξ ορισμού, δεν περιλαμβάνουν τη λειτουργία του ίδιου του υπολογιστή. Οι εφαρμογές έχουν σχεδιαστεί να τρέχουν σε συγκεκριμένες συσκευές και γράφονται για ένα συγκεκριμένο λειτουργικό σύστημα (όπως το iOS, Windows ή Android). Ακόμα και σύνθετα λογισμικά που έχουν σχεδιαστεί για χρήση σε έναν προσωπικό υπολογιστή είναι εφικτό να σχεδιαστούν σε αντίστοιχα app για μια κινητή συσκευή. Μια τέτοια εφαρμογή είναι ικανή να προσφέρει παρόμοια λειτουργικότητα σε σχέση με το πλήρες λογισμικό της που τρέχει στον υπολογιστή και βελτιστοποίηση εμφάνισης των δεδομένων, λαμβάνοντας υπόψη το μέγεθος της οθόνης και την ανάλυση.

Τα app stores συνήθως λαμβάνουν την μορφή ενός ηλεκτρονικού καταστήματος όπου οι χρήστες μπορούν να περιηγηθούν μέσα από διάφορες

κατηγορίες εφαρμογών, να δουν πληροφορίες για κάθε εφαρμογή (όπως σχόλια ή εκτιμήσεις) καθώς και να την αποκτήσουν δωρεάν ή επί πληρωμής. Η επιλεγμένη εφαρμογή προσφέρεται ως μια αυτόματη λήψη που θα εγκατασταθεί στη κινητή συσκευή(11).

Οι χρήστες smartphone αντιπροσωπεύουν σήμερα περίπου το ένα τέταρτο από τα τέσσερα δισεκατομμύρια χρήστες κινητών τηλεφώνων παγκοσμίως. Έτσι, οι βελτιστοποιήσεις στο περιεχόμενο των συσκευών καθώς και η ικανοποίηση των αναγκών των χρηστών αποτελούν ένα σοβαρό κεφάλαιο για τις εταιρείες κινητών τηλεφώνων. Οι mobile εφαρμογές που διατίθενται από τα καταστήματα εφαρμογών κάθε λογισμικού, όπως App store (iOS), Play store (Android), Windows Store, κ.α. προσφέρουν ένα γρήγορο και βολικό τρόπο για τους χρήστες να έχουν πρόσβαση σε πληροφορίες που είναι χρήσιμες για την καθημερινότητα τους. Οι καταναλωτές ξοδεύουν περισσότερο χρόνο για τις εφαρμογές τους από ποτέ. Κατά την τελευταία πενταετία οι πωλήσεις smartphone έχουν πολλαπλασιαστεί, δημιουργώντας έτσι μια κοινωνία χιλιάδων εφαρμογών και δυνατότητες για νέες ευκαιρίες σύνδεσης των εμπόρων με τους καταναλωτές. Σύμφωνα με μία μελέτη που διεξήχθη από την Flurry Analytics οι εφαρμογές δαπανούν το 86% του χρόνου του καταναλωτή. Οι χρήστες smartphone καταναλώνουν ανά ημέρα κατά μέσο όρο 2 ώρες και 42 λεπτά στις κινητές συσκευές τους, τις 2 ώρες και 19 λεπτά χρησιμοποιούν ή εκτελούν μια εργασία σε μια εφαρμογή. Το παρακάτω διάγραμμα δείχνει ότι οι χρήστες περνούν περισσότερο χρόνο σε τουλάχιστον μια εφαρμογή απ' ότι στο διαδίκτυο(12).



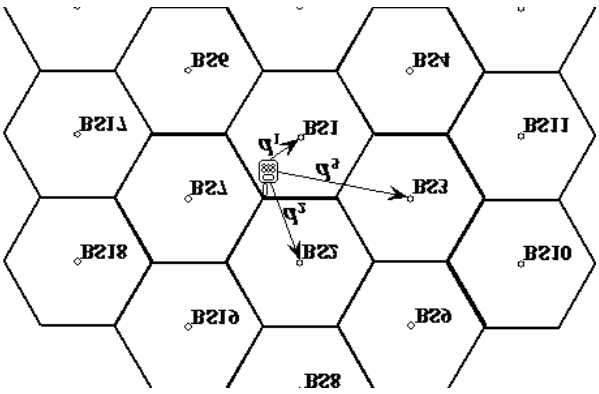
Γράφημα 1-3: Δαπάνη χρόνου χρήσης smartphone εφαρμογών/διαδικτύου το 2013-2014 (σε %) (12).

1.2. Κινητό διαδίκτυο & GPS

1.2.1. Δίκτυα κινητής τηλεφωνίας

Το κινητό διαδίκτυο αναφέρεται στην πρόσβαση του World Wide Web, δηλαδή την χρήση browser-based Internet υπηρεσιών, από μια φορητή συσκευή όπως ένα smartphone, συνδεδεμένο σε ένα δίκτυο κινητής τηλεφωνίας ή άλλων ασύρματων δικτύων. Ένα από τα πιο γνωστά παραδείγματα ασύρματης πρόσβασης στο internet είναι τα σύγχρονα δίκτυα κινητής τηλεφωνίας που χρησιμοποιούν οι εταιρίες παροχής δικτύου σε παγκόσμιο επίπεδο. Ακολούθως θα γίνει μερική ανάλυση της εξέλιξης των δικτύων κινητής τηλεφωνίας έως σήμερα.

Τα **δίκτυα πρώτης γενιάς** (1G) αναφέρονται στη πρώτη γενιά ασύρματης τηλεφωνικής τεχνολογίας. Είναι τα πρότυπα αναλογικής τηλεπικοινωνίας που εισήχθησαν τη δεκαετία του 1980 και συνεχίστηκαν μέχρι που αντικαταστάθηκαν από τα δίκτυα 2ης γενιάς ψηφιακών τηλεπικοινωνιών (2G). Η κύρια διαφορά μεταξύ των δύο συστημάτων κινητής τηλεφωνίας είναι ότι τα ραδιοσήματα που έκαναν χρήση τα δίκτυα $1^{ης}$ γενιάς ήταν αναλογικά και επέτρεπαν μόνο τη μετάδοση της φωνής και όχι την αποστολή κειμένου ή δεδομένων. Το Advanced Mobile Phone System (AMPS) είναι το πρώτο κυψελοειδές δίκτυο που υλοποιήθηκε το 1980 και παράχθηκε το 1983 στις ΗΠΑ. Η βασική ιδέα στην οποία βασίστηκε η κατασκευή των κυψελοειδών δικτύων στηρίχθηκε στην έννοια της κυψέλης, δηλαδή στην τμηματοποίηση της γεωγραφικής περιοχής που πρόκειται να καλυφθεί ηλεκτρομαγνητικά σε μικρές ζώνες, όπου υπάρχει η δυνατότητα επαναχρησιμοποίησης των ίδιων συχνοτήτων. Η σπουδαιότητα του, σε σχέση με τα παραδοσιακά συστήματα κινητής τηλεφωνίας, είναι η κάλυψη μεγαλύτερων γεωγραφικών περιοχών, η αύξηση χωρητικότητας του δικτύου, προσφορά ποιότητας στην επικοινωνία καθώς και η έλλειψη παρεμβολών ή και λανθασμένων μεταδιδόμενων πληροφοριών(13)(14).



Εικόνα 1-5: Αρχιτεκτονική κυψελοειδών δικτύων (1G) (51).

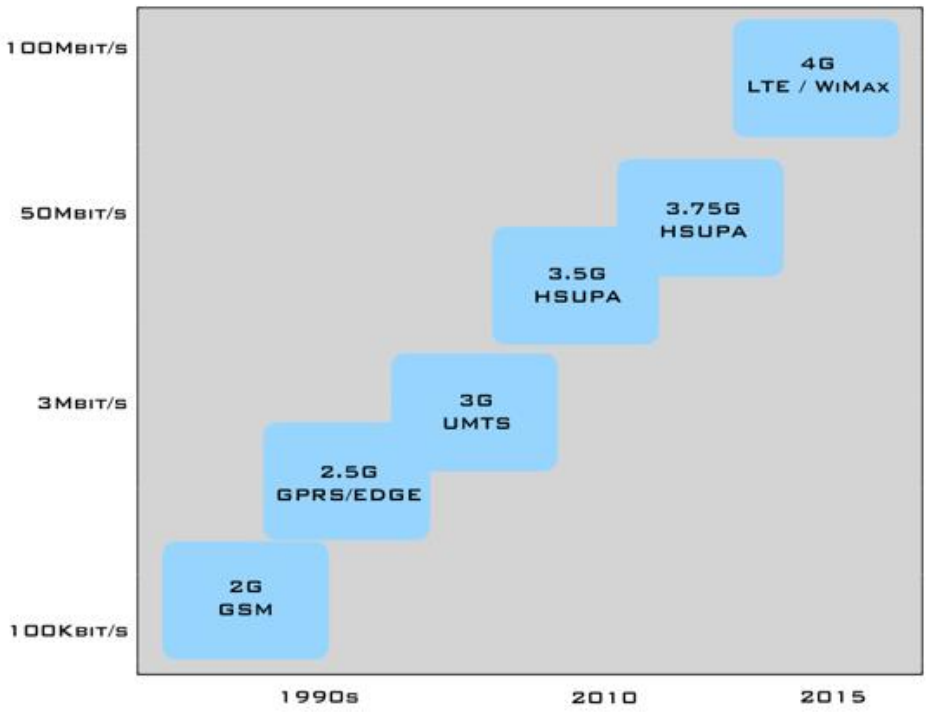
Η **δεύτερη γενιά** των δικτύων κινητής τηλεφωνίας χαρακτηρίζεται από την παροχή φωνητικών υπηρεσιών και υπηρεσιών αποστολής δεδομένων. Τα δίκτυα δεύτερης γενιάς ή 2G ξεκίνησαν την εμπορική τους λειτουργία πάνω στο πρότυπο GSM στη Φινλανδία από την Radiolinja το 1991. Το Global System for Mobile Communications (Παγκόσμιο Σύστημα Κινητών Επικοινωνιών) εν συντομία GSM σχεδιάστηκε ώστε μα εκμεταλλεύεται τη ζώνη συχνοτήτων των 900MHZ. Στην Ελλάδα η τεχνολογία του GSM χρησιμοποιήθηκε στα δίκτυα κινητής τηλεφωνίας που εγκαταστάθηκαν στις αρχές της δεκαετίας του ΄90. Το GSM είναι ένα κυψελοειδές ψηφιακό σύστημα κινητής τηλεφωνίας δεύτερης γενιάς, το οποίο χρησιμοποιεί ηλεκτρομαγνητικά σήματα και την τεχνική πολλαπλής πρόσβασης με διαχωρισμό του διαθέσιμου φάσματος συχνοτήτων σε ένα αριθμό καναλιών και την διαίρεση αυτών σε χρονοθυρίδες για την μετάδοση σημάτων. Το κύριο όφελος των δικτύων 2G σχέση με τους προκατόχους τους είναι ότι οι τηλεφωνικές συνομιλίες κρυπτογραφούνται ψηφιακά. Τα δίκτυα 2G εισήγαγαν υπηρεσίες δεδομένων κινητής τηλεφωνίας, ξεκινώντας με τα μηνύματα κειμένου SMS και μηνύματα multimedia MMS(15)(16).

Εικόνα 1-6: Λογότυπο προτύπου Global System for Mobile Communications (52).

Επόμενα είναι τα **δίκτυα 3ης γενιάς** ή 3G, που αποτέλεσαν ένα μεγάλο άλμα για την άμεση επικοινωνία. Το 3G βασίζεται στην χρήση ενός συνόλου προτύπων όπως τα WCDMA, UMTS, CDMA2000 και EDGE που χρησιμοποιούν τεχνολογίες και δίκτυα σύμφωνα με τις διεθνής κινητές τηλεπικοινωνίες. Με τα δίκτυα τρίτης γενιάς βρήκαν εφαρμογή η ασύρματη φωνητική τηλεφωνία, η κινητή και σταθερή πρόσβαση στο Internet και οι κλήσεις βίντεο. Την διαφορά πού αποτελεί παράδειγμα της κοινής πορείας μεταξύ υπολογιστών και τηλεπικοινωνιών έγινε από το Mobile Internet κάνοντας δυνατή την πρόσβαση σε δεδομένα με τη χρήση μόνο μιας κινητής συσκευής πού υποστηρίζει υπηρεσίες ταχύτητας μεταφοράς πληροφοριών το λιγότερο 200

Kbit/s. Αργότερα το 3G εξελίχθηκε σε 3.5G και 3.75G προσφέροντας κινητή ευρυζωνική πρόσβαση αρκετών Mbit/s σε smartphones και κινητά μόντεμ φορητών υπολογιστών. Κάθε γενιά χαρακτηρίζεται από νέες ζώνες συχνοτήτων, υψηλότερες ταχύτητες μετάδοσης δεδομένων και μη συμβατή τεχνολογία μετάδοσης. Μια νέα γενιά δικτύου κινητής τηλεφωνίας εμφανίζεται περίπου κάθε δέκα χρόνια από το πρώτο πρότυπο 1G που εισήχθη το 1981-1982, τα δεύτερα την δεκαετία του 90, τα δίκτυα 3G το 1998 και της τέταρτης γενιάς "4G" το 2008(17)(18).

Κύριο χαρακτηριστικό των **δικτύων 4ης γενιάς** είναι οι ιδιαίτερα γρήγορες ταχύτητες δεδομένων που φτάνουν έως το 100 Mbit/s και η ποιότητα των προσφερόμενων υπηρεσιών. Με τις μεγάλες ταχύτητες να έχουν κάνει την εμφάνιση τους ήταν δεδομένο νέες υπηρεσίες όπως το OnLine gaming μέσω κινητών συσκευών και η τηλεφωνία IP. Το πρότυπο που χρησιμοποιείται στις μέρες μας από τα δίκτυα 4ης γενιάς είναι το LTE Advanced, μία αναβάθμιση του σημερινού LTE. Το LTE Advanced προτάθηκε την άνοιξη του 2009 και αναμένεται να χρησιμοποιηθεί στις αρχές του 2016 υποστηρίζοντας πολύ μεγαλύτερες ταχύτητες. Το γεγονός ότι αποτελεί αναβάθμιση των παλιότερων δικτύων συμβάλλει στη μείωση του κόστους (19).



Γράφημα 1-4: Οι ταχύτητες δικτύων κινητής τηλεφωνίας ανά χρονολογία (σε γενιές/πρότυπα) (53).

1.2.2. Τεχνολογία Wi-Fi

Αν και έχουν αναπτυχθεί πολλές τεχνολογίες και πρότυπα για τα ασύρματα δίκτυα, κατά την δεκαετία του ΄90, μια συγκεκριμένη κατηγορία προτύπων είχε κυριαρχήσει, το ασύρματο LAN IEEE 802.11 (IEEE 802.11 Wireless LAN) γνωστό και ως Wi-Fi. Ο όρος Wi-Fi (Wireless Fidelity) χρησιμοποιείται για να προσδιορίσει τις συσκευές που βασίζονται στην προδιαγραφή IEEE 802.11 b/g/n και εκπέμπουν σε συχνότητες 2.4 GHz. Ωστόσο το Wi-Fi έχει επικρατήσει και ως όρος αναφερόμενος συνολικά στα ασύρματα δίκτυα. Συνήθεις εφαρμογές του είναι η παροχή ασύρματων δυνατοτήτων πρόσβασης στο Internet, τηλεφωνίας μέσω διαδικτύου (VoIP) και διασύνδεσης μεταξύ ηλεκτρονικών συσκευών όπως τηλεοράσεις, ψηφιακές κάμερες, DVD Player και ηλεκτρονικοί υπολογιστές. Σε φορητές ηλεκτρονικές συσκευές το 802.11 βρίσκει εφαρμογές ασύρματης μετάδοσης, όπως π.χ. στη μεταφορά φωτογραφιών από ψηφιακές κάμερες σε υπολογιστές για περαιτέρω επεξεργασία και εκτύπωση, αν και σε αυτόν τον τομέα έχει υποσκελιστεί από το πρωτόκολλο Bluetooth για τα πολύ μικρότερης εμβέλειας ασύρματα προσωπικά δίκτυα(20).

Υπάρχουν αρκετά πρωτόκολλα 802.11 για την τεχνολογία ασυρμάτων LAN, που περιλαμβάνουν τα 802.11b, 802.11a και 802.11g. Αυτά τα τρία πρότυπα έχοντας πολλά κοινά χαρακτηριστικά, χρησιμοποιούν το ίδιο πρωτόκολλο πρόσβασης μέσου CSMA/CA και την ίδια δομή πλαισίου για τα πλαίσια επιπέδου ζεύξης. Και τα τρία πρότυπα έχουν τη δυνατότητα να μειώσουν το ρυθμό μετάδοσης ώστε να φτάσουν σε μεγαλύτερες αποστάσεις. Επίσης, επιτρέπουν την λειτουργία υποδομής και τον τρόπο λειτουργίας ad hoc. Ωστόσο έχουν ορισμένες βασικές διαφορές στο φυσικό επίπεδο. Τα ασύρματα LAN 802.11b έχουν ρυθμό δεδομένων 11Mbps και λειτουργούν στη χωρίς άδεια χρήσης ζώνη συχνοτήτων 2.4-2.485 GHz, ανταγωνιζόμενο για φάσμα συχνοτήτων με τα τηλέφωνα 2.4 GHz. Τα ασύρματα LAN 802.11a μπορούν να λειτουργούν σε σαφώς υψηλότερους ρυθμούς bit, αλλά και σε υψηλότερες συχνότητες. Λειτουργώντας σε υψηλότερη συχνότητα, τα LAN 802.11a έχουν μικρότερες αποστάσεις μετάδοσης για ένα δεδομένο επίπεδο ισχύος και έχουν προβλήματα λόγω της διάδοσης πολλαπλών διαδρομών. Τα LAN 802.11g, που λειτουργούν στην ίδια χαμηλότερη ζώνη συχνοτήτων με τα LAN 802.11b είναι συμβατά προς τα πίσω με το LAN 802.11b (έτσι ώστε κάποιος να μπορεί να αναβαθμίσει αυξητικά πελάτες 802.11b), αλλά με τις υψηλότερες ταχύτητες μετάδοσης του 802.11a, επιτρέπουν στους χρήστες να έχουν τα πλεονεκτήματα και των δύο προτύπων.

Ένα νέο πρότυπο Wi-Fi, το 802.11n [IEEE 802.11n 2007], βρέθηκε στη διαδικασία προτυποποίησης. Το LAN 802.11n χρησιμοποιεί κεραίες πολλαπλών εισόδων, πολλαπλών εξόδων (multiple-output, MIMO), δηλαδή δύο ή περισσότερες κεραίες στην πλευρά αποστολής και δυο ή περισσότερες κεραίες στην πλευρά λήψης με αποτέλεσμα να μεταδίδουν και να λαμβάνουν διαφορετικά σήματα. Αν και το πρότυπο αυτό δεν είχε οριστικοποιηθεί, υπάρχουν διαθέσιμα προιόντα με βάση την υφιστάμενη κατάσταση του προτύπου και οι πρώτες δοκιμές έδειχναν ότι μπορούσε πρακτικά να επιτευχθεί διεκπεραιωτή ικανότητα πάνω από 100 Mbps(1)(20).

Το νεότερο πρότυπο IEEE 802.11ac, τροποποίηση του IEEE 802.11, που δημοσιεύθηκε τον Δεκέμβριο του 2013, βασίζεται στο πρότυπο 802.11n και αποτελεί την πιο εξελιγμένη τεχνολογία Wi-Fi ως σήμερα. Αυτή η προδιαγραφή έχει πολλαπλούς σταθμούς throughput WLAN τουλάχιστον 1 gigabit ανά δευτερόλεπτο και ένα μονό σύνδεσμο throughput τουλάχιστον 500 megabits ανά δευτερόλεπτο (500 Mbit/s). Οι αλλαγές σε σύγκριση με το 802.11n περιλαμβάνουν ευρύτερα κανάλια (80 ή 160 MHz έναντι 40 MHz) στη ζώνη των 5 GHz, περισσότερα χωρικά ρεύματα (μέχρι οκτώ έναντι τεσσάρων), διαμόρφωση υψηλότερης τάσης (έως 256-QAM vs 64-QAM), και προσθήκη πολλαπλών χρηστών MIMO (MU-MIMO). (21)(22).

Standard	Frequency Band	Max. MIMO	Max. Raw Data Rate
802.11a	5 GHz	None	54 Mbps
802.11b	2.4 GHz	None	11 Mbps
802.11g	2.4 GHz	None	54 Mbps
802.11n	2.4/5 GHz	4x4	600 Mbps
802.11ac	5 GHz	8x8	6.93 Gbps

Πίνακας 1-3: Σύνοψη των ασύρματων προτύπων IEEE 802.11 (54).

Γνωρίζουμε πως η τεχνολογία Wi-Fi έχει αναπτυχθεί και εξελιχθεί πολύ γρήγορα. Χρησιμοποιείται σε κτίρια, γραφεία, σε κοινούς χώρους ακόμα και στον καθημερινό μας εξοπλισμό όπως τα Smartphone, Laptop, Wearable συσκευές και πολλές άλλες. Ένα σημαντικό ζήτημα με τα τρέχον πρότυπα αυτής της τεχνολογίας ήταν ο τρόπος με τον οποίο οι συσκευές θα αλληλοεπιδρούν μεταξύ τους. Στις μέρες μας βέβαια το πρόβλημα αυτό έχει λυθεί έχοντας ακόμα και τα smartphone κεραία που να μπορεί επικοινωνήσει με όλα τα πρότυπα.

Εικόνα 1-7: 802.11a/b/g/n/ac Wi-Fi (55).

1.2.3. GPS

Το GPS (Global Positioning System) είναι το παγκόσμιο σύστημα προσδιορισμού θέσης. Συγκεκριμένα είναι ένα δίκτυο δορυφόρων που κινούνται σε τροχιά γύρω από τη Γη, σε σταθερά σημεία επάνω από τον πλανήτη και μεταδίδουν σήματα στη Γη προς όλους όσους διαθέτουν δέκτη GPS. Αυτά τα σήματα φέρουν δεδομένα για τον χρόνο και τις συντεταγμένες ενός σημείου στον πλανήτη παρέχοντας στους χρήστες τη δυνατότητα να εντοπίζουν την ακριβή τους θέση, την ταχύτητα και την ώρα σε οποιοδήποτε σημείο του πλανήτη(23).

Το πρώτο σύστημα GPS αναπτύχθηκε κατά τη δεκαετία του 1960 προκειμένου να παρέχει στα σκάφη του πολεμικού ναυτικού των ΗΠΑ τη δυνατότητα να διαπλέουν τους ωκεανούς με μεγαλύτερη ακρίβεια. Διέθετε πέντε δορυφόρους και παρείχε στα σκάφη τη δυνατότητα να ελέγχουν τη θέση τους ανά μία ώρα. Σήμερα, οι φορητές συσκευές δορυφορικής πλοήγησης μπορούν να παρέχουν στους οδηγούς την ακριβή θέση τους με απόκλιση λίγων μέτρων, μία επαρκώς ακριβή ένδειξη για την οδική κυκλοφορία. Οι στρατιωτικές εφαρμογές έχουν σαφώς μεγαλύτερη ακρίβεια με αποτέλεσμα μία θέση να μπορεί να εντοπισθεί με απόκλιση λίγων εκατοστών. Ωστόσο, από τις αρχές της δεκαετίας του 1980 και μετά, το GPS είναι διαθέσιμο προς χρήση σε όλους όσοι διαθέτουν ένα δέκτη GPS. Αεροπορικές εταιρείες, ναυτιλιακές εταιρείες, εταιρείες οδικών μεταφορών και οδηγοί σε οποιοδήποτε σημείο του πλανήτη χρησιμοποιούν το σύστημα GPS για να παρακολουθούν οχήματα, να ακολουθούν την καλύτερη διαδρομή που θα τους οδηγήσει το συντομότερο δυνατό από το σημείο Α στο σημείο Β(24)(25).

Η μεγάλη εξάπλωση της χρήσης του GPS οφείλεται και στη διάδοση των οικονομικά προσιτών φορητών δεκτών GPS για πεζούς ή οχήματα και των γενικών υπολογιστικών συσκευών (όπως τα smartphones και Tablet) με ενσωματωμένο δέκτη GPS. Ένας φορητός δέκτης αποτελείται από:

- Την εσωτερική δορυφορική κεραία, η οποία λαμβάνει το σήμα GPS από τους δορυφόρους με τους οποίους έχει οπτική επαφή. Επίσης, λαμβάνει σήμα και από

ανακλάσεις, π.χ. σε τοίχους, κάνοντας δυνατή τη λήψη σε δρόμους που περιβάλλονται από πολύ ψηλά κτήρια ή ακόμη και σε κάποιους εσωτερικούς χώρους. Πάντως, αρκετοί δέκτες διαθέτουν υποδοχή για εξωτερική κεραία.

- Τον κυρίως δέκτη GPS ο οποίος χρησιμοποιεί κυκλώματα εξαιρετικά χαμηλού θορύβου και ειδικές τεχνικές επεξεργασίας σήματος ώστε να ξεχωρίζει τα εξαιρετικά ασθενή σήματα από τους δορυφόρους, από τον ισχυρό τηλεπικοινωνιακό θόρυβο ο οποίος έχει τη μορφή τυχαίου σήματος. Ο κυρίως δέκτης αποτελείται από το αναλογικό τμήμα εισόδου και το ψηφιακό, το οποίο περιέχει σύνθετο ψηφιακό υλικό (hardware), συνήθως κάποιο εξειδικευμένο ολοκληρωμένο κύκλωμα τύπου ASIC και μικροελεγκτή (microcontroller) χαμηλής κατανάλωσης ισχύος(25).

1.3. Web Services

1.3.1. Τι είναι Web Services

Υπάρχουν πολλοί ορισμοί για το τι είναι Web Services. Σύμφωνα με την IBM τα Web Services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα Web Service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μια λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από Web Services οι οποίες αλληλοεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή Web Services.

Όλα τα Web Services σύμφωνα με τη Microsoft έχουν τρία κοινά χαρακτηριστικά:

- Εκθέτουν χρήσιμη λειτουργικότητα σε χρήστες του διαδικτύου μέσα από ένα πρότυπο δικτυακό πρωτόκολλο. Στις περισσότερες περιπτώσεις αυτό το πρωτόκολλο είναι το SOAP (Simple Object Access Protocol).
- Παρέχουν ένα τρόπο περιγραφής των διεπαφών τους με αρκετή λεπτομέρεια ώστε να επιτρέψουν στο χρήστη τους να χτίσει μια εφαρμογή τύπου client για να επικοινωνεί μαζί τους. Η περιγραφή συνήθως παρέχεται σε ένα έγγραφο XML το οποίο ονομάζεται έγγραφο WSLD (Web Services Description Language).
- Καταχωρούνται έτσι ώστε οι δυνητικοί χρήστες να μπορούν να τα βρούνε εύκολα. Αυτό γίνεται με το UDDI (Universal Discovery Description and Integration).

Τα web Services λοιπόν αποτελούν μία αρχιτεκτονική κατανεμημένων συστημάτων κατασκευασμένη από πολλά διαφορετικά υπολογιστικά συστήματα τα οποία επικοινωνούν μέσω του δικτύου ώστε να δημιουργήσουν ένα σύστημα. Αποτελούνται από ένα σύνολο προτύπων τα οποία επιτρέπουν στους υπεύθυνους ανάπτυξης εφαρμογών (προγραμματιστές) να υλοποιήσουν κατανεμημένες εφαρμογές που χρησιμοποιούν ένα συνδυασμό από ενότητες λογισμικού (software modules) οι οποίες καλούνται από συστήματα που ανήκουν σε διαφορετικά τμήματα ενός οργανισμού ή σε διαφορετικούς οργανισμούς(26).

1.3.2. Τεχνολογίες των Web Services

Στον παρακάτω πίνακα σημειώνονται οι διάφορες τεχνολογίες που συντελούν ένα Web Service καθώς και μία συνοπτική περιγραφή τους.

Τεχνολογία	**Επίπεδο**	**Περιγραφή**
XML	Ομοιόμορφος ορισμός και ανταλλαγή δεδομένων	Η Extended Markup Language (XML) είναι μια περιγραφική γλώσσα η οποία έχει καλή καθορισμένη σύνταξη και σημασιολογία. Τα «αυτοπεριγραφικά» χαρακτηριστικά της XML την κάνουν απλή, δυνατή και μηχανισμό για τη σύλληψη και την ανταλλαγή των στοιχείων μεταξύ των διαφορετικών εφαρμογών.
SOAP	Πρότυπο κανάλι επικοινωνίας	Το Simple Object Access Protocol (SOAP) είναι το κανάλι που χρησιμοποιείται για επικοινωνία μεταξύ μιας εφαρμογής - Server Web Service και μιας εφαρμογής - client. Η απλότητα του SOAP είναι το ότι δεν καθορίζει κανένα νέο πρωτόκολλο μεταφοράς. Αντίθετα, επαναχρησιμοποιεί μεταξύ άλλων το Hyper Text Transfer Protocol (HTTP) ή το Simple Mail Transfer Protocol

		(SMTP) για μεταφορά δεδομένων σαν μηνύματα. Αυτή η χρήση του HTTP ή του SMTP σαν πρωτόκολλο μεταφοράς εξασφαλίζει ότι οι εφαρμογές Server με τις εφαρμογές client μπορούν να επικοινωνήσουν χρησιμοποιώντας το διαδίκτυο σαν ραχοκοκαλιά. Είναι η χρήση του SOAP που πολλαπλασιάζει τις ικανότητες των web services.
WSDL	Πρότυπη περιγραφική γλώσσα για την περιγραφή των παρεχόμενων υπηρεσιών	Οι εφαρμογές που παρέχουν web services διαφημίζουν τις διάφορες υπηρεσίες που παρέχουν χρησιμοποιώντας μια πρότυπη περιγραφική γλώσσα που ονομάζεται Web Services Description Language (WSDL). Κατά τρόπο ενδιαφέροντα, η WSDL βασίζεται στην XML και χρησιμοποιεί ένα ειδικό σύνολο ετικετών (tags) για να περιγράψει ένα web service, τις υπηρεσίες που παρέχονται, που να εντοπιστεί και ούτω καθ' εξής. Οι εφαρμογές client λαμβάνουν πληροφορίες για ένα web service πριν από την πρόσβασή τους σε αυτό και τελικά τη χρήση του.
UDDI	Καταχώρηση και εντοπισμός των παρεχόμενων υπηρεσιών	Ο «χρυσός οδηγός» των web services είναι το Universal Description Discovery and Integration (UDDI). Οι εφαρμογές που παρέχουν web services παρατίθενται σε ένα κατάλογο από πάροχους υπηρεσιών χρησιμοποιώντας το UDDI. Παρόμοια,

		οι εφαρμογές client εντοπίζουν τους πάροχους εφαρμογών web services χρησιμοποιώντας UDDI. Όπως και στην περίπτωση της WSDL, και το UDDI βασίζεται στην XML.

Πίνακας 1-4: Τεχνολογίες Web Services (27).

1.3.3. Πλεονεκτήματα των Web Services

Τα Web Services μας προσφέρουν κάποια πλεονεκτήματα σε σχέση με παλαιότερες κατανεμημένες τεχνολογίες, τα οποία είναι:

- Ευκολότερος χειρισμός δεδομένων: Το κυριότερο πρόβλημα στις κατανεμημένες τεχνολογίες ήταν το λεγόμενο tight-coupling ή στα ελληνικά η ισχυρή συνδεσιμότητα. Μια εφαρμογή που καλούσε μια άλλη απομακρυσμένη ήταν αυστηρά δεμένη με αυτή από τη κλήση λειτουργίας (function call) που εκτελούσε και τις παραμέτρους που περνούσε. Στα περισσότερα συστήματα πριν από την έλευση των web services ο τρόπος επικοινωνίας ήταν μια σταθερή διεπαφή με λίγη έως καθόλου ευελιξία ή προσαρμοστικότητα στα περιβάλλοντα ή τις ανάγκες που μεταβάλλονται συνεχώς. Τα web services χρησιμοποιούν τη γλώσσα XML η οποία μπορεί να περιγράψει οποιαδήποτε δεδομένα σε ένα πραγματικά ανεξάρτητο από πλατφόρμα τρόπο για ανταλλαγή αυτών των δεδομένων μεταξύ συστημάτων. Με αυτόν τον τρόπο οδηγούμαστε σε εφαρμογές με χαλαρή συνδεσιμότητα (loosely-coupled). Επιπλέον τα web services μπορούν να λειτουργήσουν σε πιο αφηρημένο επίπεδο στο οποίο μπορούν να επαναξιολογήσουν, να τροποποιήσουν ή να χειριστούν τύπους δεδομένων δυναμικά κατά περίπτωση. Έτσι σε τεχνικό επίπεδο τα web services μπορούν να χειριστούν δεδομένα πολύ ευκολότερα και να επιτρέψουν στο λογισμικό να επικοινωνεί πιο ελεύθερα.
- Απλότητα πρωτοκόλλου επικοινωνίας: Τα web services χρησιμοποιούν ως πρωτόκολλο επικοινωνίας το SOAP. Το πρωτόκολλο αυτό είναι πολύ πιο απλό από πρωτόκολλα παλαιότερων τεχνολογιών όπως αυτά που χρησιμοποιούνταν από τα κατανεμημένα περιβάλλοντα CORBA , DCOM, RPC. Έτσι το να δημιουργήσει κανείς μια υλοποίηση SOAP που υπόκειται στα πρότυπα (standards-compliant) είναι πολύ πιο εύκολο. Σήμερα μπορεί να βρει κανείς υλοποιήσεις του SOAP από

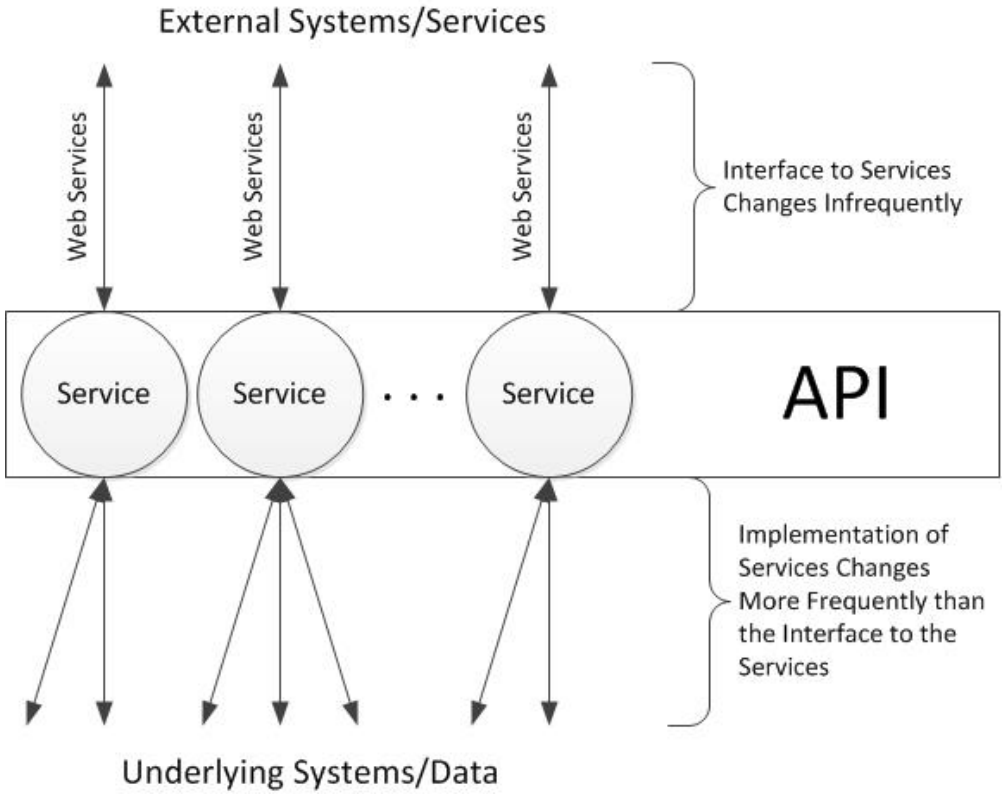
τις μεγαλύτερες εταιρίες πληροφορικής αλλά ακόμη και από μεμονωμένους προγραμματιστές, πράγμα αδιανόητο για παλαιότερες κατανεμημένες τεχνολογίες.

- Απλότητα υποδομής: Τα web services λειτουργούν με πρότυπες γλώσσες και πρωτόκολλα όπως η XML , το HTTP και το TCP/IP. Η πλειονότητα των εταιριών έχουν ήδη την δικτυακή υποδομή και τους ανθρώπους με γνώσεις και εμπειρία που τη συντηρούν. Έτσι το κόστος για την εφαρμογή των web services είναι σημαντικά μικρότερο από αυτό των προηγούμενων τεχνολογιών.
- Ευκολία στην επικοινωνία: Με τις προηγούμενες τεχνολογίες η συνεργασία μεταξύ εταιριών ήταν ένα θέμα διότι κατανεμημένες τεχνολογίες όπως CORBA και DCOM χρησιμοποιούσαν μη πρότυπες πόρτες. Σαν αποτέλεσμα η συνεργασία σήμαινε άνοιγμα "οπών" στα τείχη προστασίας (firewalls) κάτι που πολλές φορές δεν ήταν αποδεκτό από τους ανθρώπους της πληροφορικής σε μια εταιρία αφού έθετε σε κίνδυνο στην ασφάλεια των συστημάτων. Το γεγονός αυτό δεν επέτρεπε δυναμική συνεργασία λόγω του ότι απαιτούσε μια χειροκίνητη διαδικασία για τη συνεργασία μιας εταιρίας με τους συνεργάτες της. Τα web services μπορούν να χρησιμοποιήσουν (μεταξύ άλλων) το HTTP ως πρωτόκολλο μεταφοράς και τα περισσότερα τείχη προστασίας επιτρέπουν την πρόσβαση μέσω της θύρας 80 (πρότυπη θύρα για το HTTP). Με αυτόν τον τρόπο οδηγούμαστε σε ευκολότερες και δυναμικές συνεργασίες μεταξύ των συστημάτων των εταιριών.
- Διαλειτουργικότητα και ευκολία ανάπτυξης νέων εφαρμογών: Οι προηγούμενες κατανεμημένες τεχνολογίες υπέφεραν από ζητήματα διαλειτουργικότητας διότι κάθε κατασκευαστής (Vendors) υλοποιούσε το δικό του πρότυπο για distributed object messaging. Με την XML σαν το μόνο πρότυπο στα web services, συστήματα φτιαγμένα από διαφορετικές τεχνολογίες όπως η Java και το .NET μπορούν να επικοινωνήσουν μεταξύ τους. Επιπλέον λόγω της απλότητας της XML είναι πολύ πιο εύκολο να γραφτούν νέες εφαρμογές σε μικρό χρονικό διάστημα(26).

1.4. Διεπαφές προγραμματισμού εφαρμογών (APIs)

Μια Διεπαφή Προγραμματισμού Εφαρμογών (API Application Programming Interface) ή αλλιώς Διασύνδεση Προγραμματισμού Εφαρμογών είναι το γραφικό περιβάλλον (Interface) που ένα υπολογιστικό σύστημα, βιβλιοθήκη ή διαδικτυακή εφαρμογή/υπηρεσία παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων.

Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το σύνολο των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή, ένα «συμβόλαιο κλήσης» μεταξύ καλούντος και καλούμενου, διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους. Το API απλά ορίζει με ποιές εξωτερικές εντολές θα παρέχει την αμφίδρομη επικοινωνία με την υπηρεσία που θέλει να συνδεθεί προς αυτό χωρίς να αποκαλύπτει τον πηγαίο κώδικά του(28)(29).



Εικόνα 1-8: Απεικόνιση API για Web Services (server-oriented architecture) (56).

ΚΕΦΑΛΑΙΟ 2. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝ WINDOWS

2.1. Η ιστορία των Windows Phone

Τα Windows Phone είναι μία πολύ γνωστή ιδέα που έχει αναπτυχθεί και βελτιωθεί κατά καιρούς από την Microsoft. Ο προκάτοχός του, το Windows Mobile ήταν ένα λειτουργικό σύστημα που αναπτύχθηκε το 2000 για Pocket PC και ονομάστηκε Pocket PC 2000 ή αλλιώς «Rapier». Μετά από αυτό το πρώτο λειτουργικό σύστημα η Microsoft ανέπτυξε διάφορα άλλα λειτουργικά συστήματα για έξυπνα κινητά τηλέφωνα και υπολογιστές τσέπης. Η τελευταία απόπειρα αυτής της σειράς λειτουργικών συστημάτων ήταν το Windows Mobile 6.5 το οποίο παρουσιάστηκε στο Παγκόσμιο Συνέδριο Κινητής Τηλεφωνίας το 2009 και κυκλοφόρησε το ίδιο έτος.

Στη συνέχεια, η Microsoft χρειάστηκε να δημιουργήσει κάτι φιλικό προς τον χρήστη. Αποτέλεσμα αυτής της προσπάθειας ήταν τα Windows Phone 7 τα οποία ανακοινώθηκαν στις 15 Φεβρουαρίου 2010, αλλά κυκλοφόρησαν επίσημα μετά από 9 μήνες, στις 8 Νοεμβρίου 2010. Το Μάιο του 2011 κυκλοφόρησε μια ενημερωμένη έκδοση που ονομάζεται Windows Phone 7.5 - Mango. Αυτή η νέα ενημερωμένη έκδοση είχε κάποια νέα χαρακτηριστικά, όπως Internet Explorer 9, πρόσβαση στο Windows Live, στο SkyDrive και στο Twitter. Το 2012 μια άλλη ενημέρωση που ονομάζεται Tango κυκλοφόρησε. Στη συνέχεια, το 2013, η Microsoft ανακοίνωσε Windows Phone 7.8. Ήταν ένας συνδυασμός των Windows Phone 7 και 8. Μερικά από τα χαρακτηριστικά του όπως η οθόνη εκκίνησης και η επιλογή κλειδώματος της οθόνης ήταν από τα Windows Phone 8. Ωστόσο, το Windows Phone 7.8 δεν ήταν αναβαθμίσιμο λόγω προβλημάτων στο λογισμικό του σύστημα.

Το Windows Phone 8 γνωστό και ως «Apollo» είναι η τελευταία έκδοση που παρουσίασε η Microsoft το 2012. Όπως και ο προκάτοχός του, το Windows Phone 8 βασίζεται επίσης στην αρχιτεκτονική CE. Ωστόσο καμία από τις προηγούμενες εκδόσεις δεν μπορούν να αναβαθμιστούν σε Windows Phone 8. Αυτά τα τηλέφωνα εμφανίστηκαν στην αγορά με μεγαλύτερες οθόνες και με προηγμένους multi core επεξεργαστές. Τα Windows Phone 8 διαθέτουν ένα μοναδικό σχεδιασμό στην αρχική σελίδα με τα εκ νέου αρκετά μεγάλα πλακίδια (Live Tiles). Ο χρήστης μπορεί να επιλέξει ένα μέγεθος, να διαγράψει ή να αλλάξει το κάθε πλακίδιο σύμφωνα με τις προτεραιότητές του. Πρωτεύον πρόγραμμα περιήγησης στο Web είναι ο Internet

Explorer 10. Τέλος, το πληκτρολόγιο που χρησιμοποιείται στα Windows Phone 8 έχει πολλές νέες χρήσιμες λειτουργίες που διευκολύνουν τον χρήστη στη σύνταξη κειμένου (30).

2.2. Λειτουργικό σύστημα Windows Phone 8.1

Τα Windows Phone 8.1 που λάμβαναν εφαρμογή σε Η/Υ, Tablet και smartphones κυκλοφόρησαν στην αγορά στις 17 Οκτωβρίου 2013 σαν αναβάθμιση των Windows Phone 8 και αποτελούν το τελευταίο λειτουργικό σύστημα της Microsoft μέχρι σήμερα. Χαρακτηριστικά του Windows Phone 8.1 είναι:

- **Cortana**: Η Cortana βοηθάει το χρήστη στην εκτέλεση κλήσεων, αποστολή κειμένων, αποθήκευση σημειώσεων, ρύθμιση υπενθυμίσεων και ειδοποιήσεων καθώς και στην άμεση ενημέρωση πληροφοριών του χρήστη (όπως καιρός, ειδήσεις και ΜΜΜ). Τέλος, εξυπηρετεί στην αναζήτηση όχι μόνο στο κινητό αλλά και στο διαδίκτυο.
- **Start Screen Splendour**: Με την αναβάθμιση του λογισμικού σε 8.1 γίνεται αντιληπτό αμέσως στην αρχική οθόνη η αλλαγή που υπάρχει στα Live Tiles τα οποία πλέον μπορούν να υποστηρίξουν μια φωτογραφία σαν ταπετσαρία στην αρχική οθόνη. Τα Live Tiles ενώνονται μεταξύ τους σαν παζλ ώστε να εμφανιστεί ολόκληρη η φωτογραφία που έχει επιλέξει ο χρήστης.
- **Εφαρμογές Ψυχαγωγίας**: Με το Xbox Video App ο χρήστης μπορεί να παρακολουθήσει, αγοράσει ή να αναζητήσει ταινίες και αγαπημένες σειρές. Το νέο Podcasts κάνει τη λειτουργία αναζήτησης και σχολιασμού εύκολη. Τέλος, με το Xbox Music App ο χρήστης έχει πληθώρα επιλογών όσον αφορά τις μουσικές του προτιμήσεις.
- **Ένας νέος τρόπος πληκτρολόγησης**: Σύροντας το δάχτυλο στο πληκτρολόγιο οθόνης ο χρήστης μπορεί να γράψει εύκολα, γρήγορα και ακριβείς κείμενα.
- **Bing apps**: Στα Windows Phone 8.1 όλες οι Bing εφαρμογές είναι προ εγκατεστημένες από την αρχή μεταξύ των οποίων είναι: Bing Finance, Bing News, Bing Sports, Bing Travel και Bing Weather.
- **Αυτόματη ενημέρωση εφαρμογών**: Στα Windows 8.1, θα δίνεται η δυνατότητα στο χρήστη για αυτόματη ενημέρωση εφαρμογών η οποία θα μπορεί να γίνεται μόνο όταν υπάρχει σύνδεση σε δίκτυο Wi-Fi.
- **Ειδοποιήσεις και ρυθμίσεις**: Μια νέα λειτουργία έκανε την εμφάνιση της με το Windows Phone 8.1. Ο χρήστης μπορεί πλέον ανά πάσα στιγμή να κατεβάσει το

παράθυρο κέντρου δράσης από την κορυφή της οθόνης όπου μπορεί γρήγορα να δει ενημερώσεις και να ενεργοποιήσει ή όχι το Wi-Fi, το Bluetooth και άλλα.

- **Καλύτερο browsing**: Με την αναβάθμιση του Internet Explorer περιλαμβάνεται η προσωπική περιήγηση, η λειτουργία ανάγνωσης και νέες ρυθμίσεις ώστε να βοηθήσουν το χρήστη να συγκεντρώσει τις πληροφορίες που επιθυμεί να περιηγηθεί. Επιπλέον, οι αποθηκευμένες καρτέλες μπορούν να μεταφερθούν αυτόματα στο Windows PC ή στο Windows Tablet.
- **Windows Phone Store**: Το Windows Phone Store προσφέρει στο χρήστη προτεινόμενες και προσωπικές εφαρμογές. Ενώ οι εφαρμογές μπορούν να ενημερωθούν αυτόματα μόνο με σύνδεση σε ένα Wi-Fi δίκτυο και η διαδικασία εγκατάστασης γίνεται ορατή στο Store App.
- **Οργάνωση αρχικής οθόνης**: Είναι δυνατή η δημιουργία, μετονομασία και αλλαγή μεγέθους (Tiles) φακέλων στην αρχική οθόνη καθώς και η εισαγωγή ενός φακέλου μέσα σε έναν άλλο.
- **Οθόνη κλειδώματος**: Η οθόνη κλειδώματος εμφανίζει εκτός από την ημερομηνία και ώρα, ειδοποιήσεις και πληροφορίες από τις εφαρμογές που έχει επιλέξει ο χρήστης της κινητής συσκευής ώστε να γνωρίζει πάντα τι νέο υπάρχει χωρίς να χρειαστεί η είσοδος στις συγκεκριμένες εφαρμογές(31)(32).

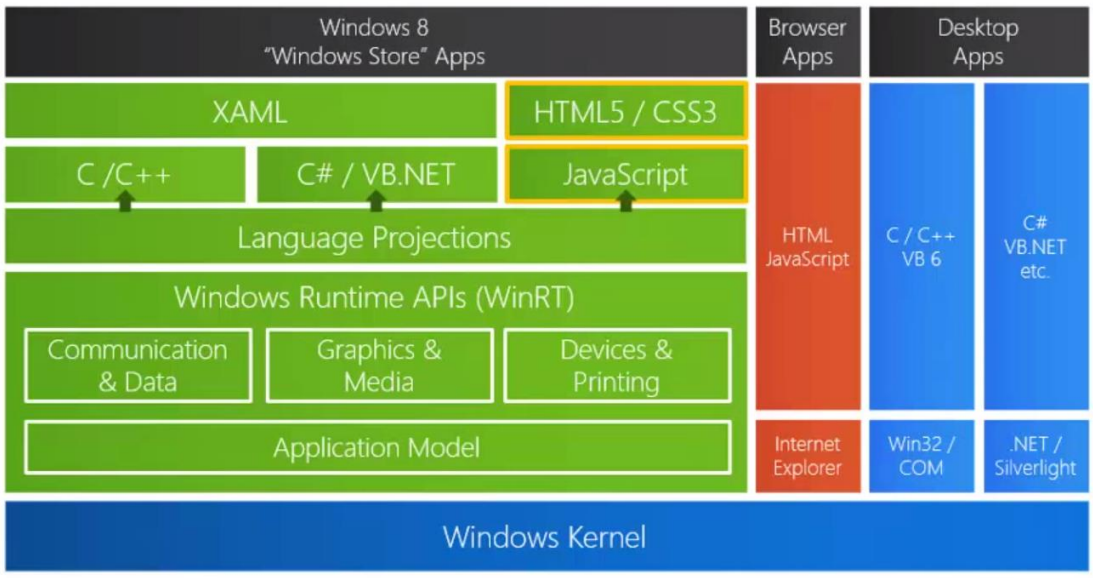
2.3. Αρχιτεκτονική του λειτουργικού Windows 8

Ξεκινώντας από το Windows Kernel, το πιο χαμηλό επίπεδο της αρχιτεκτονικής του λειτουργικού Windows 8, θα αναλύσουμε τα είδη εφαρμογών που μπορεί να αναπτύξει ένας προγραμματιστής. Μια πρώτη βασική κατηγορία είναι οι Desktop εφαρμογές, όπου μπορούν να αναπτυχθούν είτε με τις γλώσσες προγραμματισμού C/C++ και VB 6 σε συνεργασία με τα Win32/com APIs είτε με τις γλώσσες C#, VB.net, κ.α. σε συνεργασία με τα .NET/Silverlight APIs. Ένα άλλο είδος είναι οι browser εφαρμογές, εκτελούνται στο φυλλομετρητή του λογισμικού (Internet Explorer) και οι γλώσσες ανάπτυξης μιας τέτοιας εφαρμογής είναι η HTML και η JavaScript. Εφαρμογές όπως το Bing Maps και το Office 360 είναι σημαντικά παραδείγματα μιας browser based εφαρμογής.

Με τη τελευταία έκδοση του λογισμικού Windows η Microsoft έδωσε την δυνατότητα δημιουργίας εφαρμογών που θα είναι συμβατές όχι μόνο με τους Intel επεξεργαστές που είχαν στο παρελθόν αλλά και με μια νέα οικογένεια συσκευών με αρχιτεκτονική συνόλου εντολών ARM σε συνεργασία με επεξεργαστές που θα είναι

επιλογή του καταναλωτή. Η τεχνολογία ARM είναι μια αρχιτεκτονική συνόλου εντολών RISC των 32-bit όσον αφορά τους επεξεργαστές, τα αρχικά σημαίνουν προχωρημένη μηχανή RISC (Advanced RISC Machine) και είναι κατάλληλα για εφαρμογές χαμηλής ισχύος(33). Για να το επιτύχει αυτό και να είναι σε θέση να αποκτήσει νέα πλεονεκτήματα όπως αυτό της αφής δημιούργησε ένα καινούργο είδος εφαρμογών που ονομάστηκε Windows Store Apps και έχει νέες κλάσεις για τις διάφορες λειτουργίες μιας εφαρμογής. Επίσης η σύνδεση του επιπέδου Windows Kernel με το πιο πάνω επίπεδο στην αρχιτεκτονική του λογισμικού υλοποιήθηκε με το νέο περιβάλλον εκτέλεσης Windows Runtime APIs ή WinRT.

Το WinRT συνδέει όλες τις απαραίτητες λειτουργίες που θα είχε ένα μοντέλο εφαρμογής όπως τον κύκλο ζωής μιας εφαρμογής, την επικοινωνία με κάποιο δίκτυο, την πρόσβαση στα δεδομένα της συσκευής, την δυνατότητα αναπαραγωγής γραφικών και άλλων πολυμέσων, την σύνδεση με άλλες συσκευές, εκτύπωση, κ.α. Σε αντίθεση με τα άλλα είδη εφαρμογών στις Windows Store εφαρμογές δημιουργήθηκε ένα νέο επίπεδο, το Language Projections που σου δίνει την συμβατή σύνταξη γλώσσας του πάνω επιπέδου, για την άμεση πρόσβαση στο κάτω επίπεδο WinRT. Αυτό σημαίνει ότι έχεις φυσική πρόσβαση στο WinRT μέσω των γλωσσών προγραμματισμού C# / VB.NET ή C / C++ και γλώσσα σήμανσης την XAML ή με την γλώσσα προγραμματισμού JavaScript και γλώσσα σήμανσης τις HTML και CSS(47).



Εικόνα 2-1: Αρχιτεκτονική του λειτουργικού συστήματος Windows 8 (47).

2.4. Πλεονεκτήματα Windows Store εφαρμογών

Τα πλεονεκτήματα των Windows Store εφαρμογών αρχίζουν να γίνονται πιο ξεκάθαρα με το πέρασμα των χρόνων σύμφωνα με τις απαιτήσεις που έχει η Microsoft απέναντι στους καταναλωτές τις. Αυτά είναι:

- Εκτελούνται σε όλες τις συσκευές. Συστήματα που έχουν ως λειτουργικό το Windows 8 και Intel επεξεργαστές μπορούν να εκτελέσουν κάθε είδος εφαρμογής σε αντίθεση με αυτά που έχουν το νέο περιβάλλον εκτέλεσης WinRT και ARM επεξεργαστές όπως είναι τα tablet και smartphones τα οποία μπορούν να εκτελέσουν μόνο Windows Store εφαρμογές.
- Δυνατότητα πώλησης της εφαρμογής στο Windows Store. Ένας προγραμματιστής μπορεί να υλοποιήσει την ιδέα του με μια εφαρμογή σε Windows και μάλιστα μέσα από τα νέα πρότυπα εφαρμογών μπορεί να φτιάξει μία εφαρμογή συμβατή με laptop, smartphone και tablet ταυτόχρονα, και να την πουλήσει.
- Μαζί με τις λειτουργίες μιας εφαρμογής υπάρχει η δυνατότητα συνεργασίας επιπρόσθετων λειτουργιών (contracts) όπως η αναζήτηση, κοινοποίηση, εκτύπωση, πρόσβαση στα αρχεία του συστήματος, κ.α.
- Υποστηρίζει διαφορετικές καταστάσεις προβολής της εφαρμογής (view states) και αυτές είναι σε πλήρης οθόνη (Full),γέμισμα της οθόνης (Fill) και καρφίτσωμα στην άκρη της οθόνη (Snap).
- Τέλος έχει βελτιωμένο κύκλο ζωής εφαρμογής (app lifecycle) και πολλές άλλες νέες δυνατότητες(47).

2.5. Τεχνολογίες για την υλοποίηση της εφαρμογής

2.5.1. Εισαγωγή

Η JavaScript γρήγορα γίνεται μια γλώσσα που μπορεί να προσαρμοστεί και να χρησιμοποιηθεί για cross-platform web based εφαρμογές ή σε μια windows store εφαρμογή σε συνεργασία με το API του λειτουργικού συστήματος. Τα Windows Phone 8.1 μας επιτρέπουν να δημιουργήσουμε μια Windows Store εφαρμογή σε γλώσσα προγραμματισμού JavaScript με αποτέλεσμα ο ίδιος κώδικας να μπορεί να τρέξει τόσο σε ένα smartphone όσο και σε έναν σταθερό υπολογιστή ή tablet. Η επιλογή της γλώσσας αυτής για την υλοποίηση της εφαρμογής 4Seasons χαρακτηρίζεται από τις πολλές νέες δυνατότητες των αντικειμένων της βιβλιοθήκης WinJS, την δημιουργία νέων αντικειμένων από τον ίδιο τον προγραμματιστή και την ταχύτητα ανταπόκρισης

της. Η βιβλιοθήκη WinJS στις μέρες δίνει τη δυνατότητα ανάπτυξης μιας εφαρμογής για κινητό τηλέφωνο χωρίς περιορισμούς παρέχοντας έτοιμα αντικείμενα όπως λίστες, γραμμές αναζήτησης, κλήσεις σε web service κ.α. Τέλος, επιτρέπει σε έναν μεγάλο αριθμό JavaScript προγραμματιστών να αναπτύσσουν εφαρμογές για Windows Phone(34)(35).

2.5.2. Κύκλος ζωής της εφαρμογής

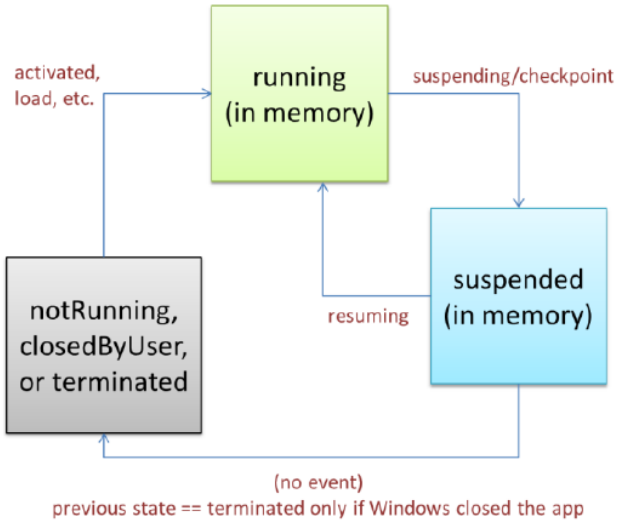
Ο κύκλος ζωής μιας εφαρμογής (application lifecycle) είναι ένα από τα βασικά χαρακτηριστικά που πρέπει να θέσει σωστά ο προγραμματιστής στην εφαρμογή του. Πέρα από το concept της ομαλής ενεργοποίησης και χρήσης της εφαρμογής πρέπει να προγραμματιστεί στο ποια θα είναι η συμπεριφορά της έξυπνης εφαρμογής όταν αντιδράσει στις επιλογές του χρήστη. Υπάρχουν διάφορες καταστάσεις λειτουργίας, όπου κάποιες είναι προγραμματισμένες η εφαρμογή να "τρέχει" κανονικά ενώ η ορατότητά της έχει χαθεί (run in background) και κάποιες άλλες μη προγραμματισμένες, για παράδειγμα ο χρήστης να τερματίσει την εφαρμογή από τον πίνακα διαχείρισης εργασιών.

Τρείς κρίσιμες καταστάσεις λειτουργίας στον κύκλο ζωής μια εφαρμογής είναι οι εξής:

- **Κατάσταση αναμονής:** Όταν η εφαρμογή δεν είναι ορατή σε καμία από τις καταστάσεις προβολής (view states), θα μπει στην κατάσταση αναμονής μετά από 5 δευτερόλεπτα με σκοπό την διατήρηση ισχύος της μπαταρίας. Αυτό σημαίνει ότι παραμένει εξ' ολοκλήρου στην μνήμη αλλά δεν κάνει χρήση του επεξεργαστή, επομένως δεν έχει δίκτυο ούτε μπορεί να υπάρξει δραστηριότητα στον σκληρό δίσκο. Όταν αυτό συμβεί, η εφαρμογή εκτελεί ένα συμβάν όπου την θέτει σε κατάσταση αναμονής (onsuspending event), αν δεν επιστρέψει εντός 5 δευτερολέπτων στην προηγούμενη κατάστασή της, το λειτουργικό σύστημα Windows θα την τερματίσει. Κατά την διάρκεια αυτής της περιόδου η εφαρμογή σώζει την περιοδική της κατάσταση και απελευθερώνει όλους του πόρους της συσκευής που έκανε χρήση.
- **Συνέχιση:** Όταν ο χρήστης συνεχίσει την λειτουργία μιας εφαρμογής που βρισκόταν σε κατάστασης αναμονής εκτελείται το συμβάν συνέχιση της εφαρμογής (onresuming event). Αυτό το συμβάν δεν προκύπτει από την ίδια την εφαρμογή γιατί σε κατάσταση αναμονής ο κώδικας της εφαρμογής δεν εκτελείται παρόλα αυτά μπορεί να γίνει έχοντας προγραμματίσει μια ουρά συμβάντων. Να

σημειώσουμε πως κατά την εκτέλεση του συμβάντος “συνέχιση της εφαρμογής”, προγραμματίζεται και για την ανανέωση του περιεχομένου της αν έχει περάσει αρκετή ώρα από τότε που τέθηκε σε αναμονή.

- **Τερματισμός:** Όταν μια εφαρμογή βρίσκεται σε κατάσταση αναμονής μπορεί να τερματιστεί αν υπάρχει ανάγκη για μνήμη. Αυτό δεν συμβαίνει μέσω κάποιου συμβάντος επειδή εξ’ ορισμού η εφαρμογή είναι ήδη σε αναμονή και ο κώδικας δεν μπορεί να εκτελεστεί. Παρόλα αυτά είναι μία από τις πιο σημαντικές στιγμές του κύκλου ζωής της εφαρμογής διότι επηρεάζει την προηγούμενη αποθηκευμένη κατάσταση που είχε για την επανεκκίνησή της (2).



Εικόνα 2-2: Process lifecycle and previousExecutionState values (2).

Ενώ έχουμε αναφέρει παραπάνω τις διάφορες καταστάσεις που μπορεί να βρεθεί μια έξυπνη εφαρμογή ας δούμε πως συνδέονται αυτά τα συμβάντα με την εκτέλεση τους και με ποιους τρόπους η εφαρμογή διαχειρίζεται την προηγούμενη της αποθηκευμένη κατάσταση (previousExecutionState) κατά την επανεκκίνησή της. Για λόγους δυσκολίας επεξήγησης των τιμών προηγούμενων καταστάσεων που μπορεί να πάρει μία εφαρμογή δημιουργήθηκε ο παρακάτω πίνακας:

Τιμές κατάστασης προηγούμενης εκτέλεσης (previousExecutionState)	Σενάρια
Not Running	Πρώτη εκτέλεση μετά την εγκατάσταση από το κατάστημα εφαρμογών (Windows Store).

	Πρώτη εκτέλεση μετά την επανεκκίνηση ή την αποσύνδεση του χρήστη. Η εφαρμογή ξεκίνησε την λειτουργία της μέσα σε 10 δευτερόλεπτα από το κλείσιμό της μέσω του χρήστη. Η εφαρμογή τερματίστηκε από την διαχείριση εργασιών ενώ λειτουργούσε ή έκλεισε από το ίδιο το λογισμικό.
Running	Η εφαρμογή λειτουργεί και καλείται να ανοίξει με έναν διαφορετικό τρόπο από αυτόν του βασικού πλακιδίου αλλά μέσω κάποιου άλλου τρόπου όπως αυτόν της αναζήτησης, κοινοποίησης, δευτερεύων πλακιδίου, μέσω ειδοποίησης, κ.α. Όταν η εφαρμογή εκτελείται και ο χρήστης πατά το πλακίδιο της εφαρμογής, τα Windows σε μεταφέρουν στην ήδη εκτελούμενη εφαρμογή χωρίς να εκτελούντα τα συμβάντα ενεργοποίησης της εφαρμογής.
Suspended	Η εφαρμογή είναι ήδη σε κατάσταση αναμονής και καλείται πατώντας ο χρήστης το βασικό πλακίδιο (όπως παραπάνω για την κατάσταση running). Σε αντίθεση με τα συμβάντα εστίασης/ορατότητας (focus/visibility events) η εφαρμογή θα λάβει συμβάν συνέχισης της εφαρμογής.
Terminated	Η εφαρμογή είχε τεθεί σε κατάσταση αναμονής και τερματίστηκε από τα Windows λόγω της έλλειψης πόρων. Σημείωση, δεν θα λάβει την εντολή του λογισμικού για τον τερματισμό της εφαρμογής (MSApp.terminateApp) διότι η εφαρμογή θα πρέπει ήδη να εκτελείται για να γίνει κλήση σε μία λειτουργία (function).
Closed by User	Η εφαρμογή έκλεισε από μία κίνηση που δεν μπορεί να διακοπεί (σύροντας το παράθυρο της εφαρμογής προς τα κάτω ή Alt + F4).

	(Ένα τερματισμός που μπορεί να διακοπεί είναι για παράδειγμα όταν ο χρήστης επιστρέφει στην εφαρμογή μέσα σε 10 δευτερόλεπτα, όπου η προηγούμενη κατάσταση της εφαρμογής θα είναι not running.)

Πίνακας 2-1: Τιμές κατάστασης προηγούμενης εκτέλεσης(2).

2.5.3. JavaScript

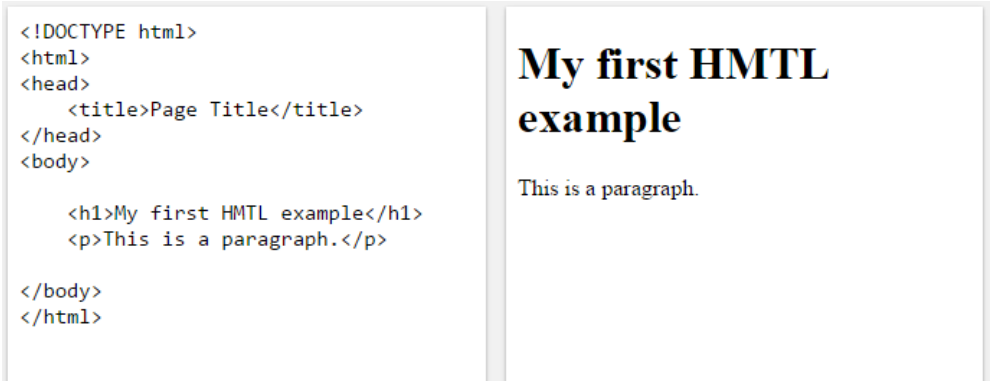
Η JavaScript είναι μια διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα βασισμένη στα πρωτότυπα γλωσσών script (prototype-based scripting language), είναι δυναμική και διαθέτει first-class συναρτήσεις. Πέρα από κάποιες ομοιότητες της JavaScript με τη Java οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων, τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers), οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets) και οι εφαρμογές για smartphone(36).

Εικόνα 2-3: Παράδειγμα γλώσσας προγραμματισμού JavaScript

2.5.4. HTML5

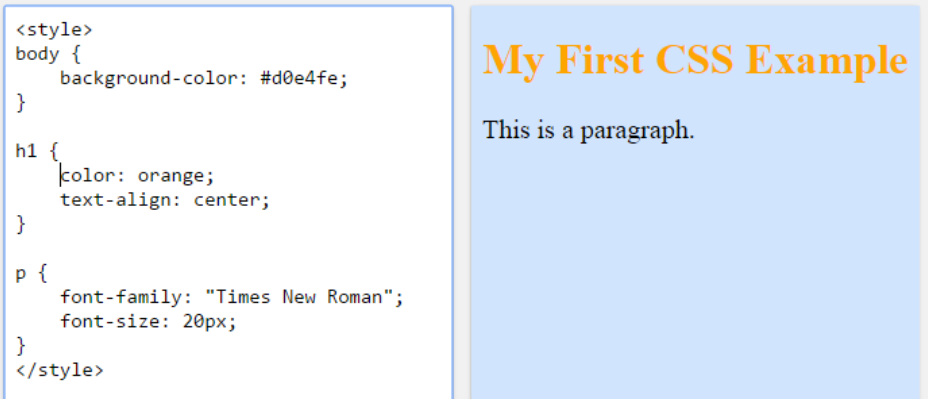
Τα αρχικά HTML σημαίνουν Γλώσσα Σήμανσης Υπερκειμένου (Hypertext Markup Language), είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML βασίστηκε στην πρότυπη γενική γλώσσα σήμανσης SGML, ένα πολύ μεγαλύτερο και πολύ πιο σύνθετο σύστημα επεξεργασίας εγγράφων. Στην ουσία η HTML, λόγω της φύσης της ως απογόνου της SGML, είναι μια γλώσσα για την περιγραφή της δομής ενός εγγράφου και όχι της παρουσίασης του. Η τελευταία έκδοση είναι η HTML5 και επιπρόσθετα καθορίζει την σήμανση στις διεπαφές προγραμματισμού εφαρμογών (APIs) που χρησιμοποιούν την γλώσσα προγραμματισμού JavaScript(37).



Εικόνα 2-4: Παράδειγμα γλώσσας σήμανσης υπερκειμένου HTML

2.5.5. CSS3

Η CSS (Cascading Style Sheets - Διαδοχικά Φύλλα Στυλ) είναι μια γλώσσα υπολογιστή, ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου γραμμένο σε μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την HTML. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη. Η τελευταία έκδοση είναι η CSS3 με πολλές επιπρόσθετες δυνατότητες(38).



Εικόνα 2-5: Παράδειγμα γλώσσας μορφοποίησης CSS

2.5.6. JSON

Η γλώσσα JSON (JavaScript Object Notation) είναι ένα ανοιχτό πρότυπο που χρησιμοποιείτε από τον άνθρωπο για να μεταδώσει αντικείμενα δεδομένων που αποτελείται από ζεύγη χαρακτηριστικών – τιμών. Χρησιμοποιείται κυρίως για τη μετάδοση δεδομένων μεταξύ του Server και Web εφαρμογής, ως εναλλακτική λύση για την XML. Παρά το γεγονός ότι προέρχονται αρχικά από τη γλώσσα JavaScript, η JSON είναι μια γλώσσα ανεξάρτητη από τη μορφή των δεδομένων και αρχικά την είχε προτείνει ο Daklas Crockford. Ο κώδικας για την ανάλυση και την παραγωγή δεδομένων JSON είναι εύκολα διαθέσιμο σε πολλές γλώσσες προγραμματισμού.. Η είσοδος στα αντικείμενα του JSON γίνεται με τον χαρακτήρα της τελείας, κάθε φορά που θέλουμε να μπούμε σε ένα εμφωλευμένο αντικείμενο και με τον αριθμό της εγγραφής μέσα σε τετράγωνες αγκύλες για την επιλογή συγκεκριμένης εγγραφής σε κάποιον πίνακα. Οι βασικοί τύποι δεδομένων ενός JSON αρχείου είναι:

1. Number: ένας δεκαδικός αριθμός που μπορεί να περιέχει κλασματικό μέρος και να χρησιμοποιεί το εκθετικό E. Η JSON δεν επιτρέπει μη φυσικούς αριθμούς, όπως NaN, ούτε κάνει οποιαδήποτε διάκριση μεταξύ ακεραίου και δεκαδικού.
2. String: μια ακολουθία από μηδέν ή περισσότερους Unicode χαρακτήρες. Τα δεδομένα τύπου string οριοθετούνται με διπλά εισαγωγικά και υποστηρίζει σύνταξη διαφυγής με κάθετο.
3. Boolean: Τύποι δεδομένων που παίρνουν ως τιμές αληθής ή ψευδείς.
4. Array: μια ταξινομημένη λίστα από μηδέν ή περισσότερες τιμές, οποιουδήποτε τύπου δεδομένων. Οι πίνακες χρησιμοποιούν την τετράγωνη αγκύλη σαν συμβολισμό των στοιχείων και κόμμα για τον διαχωρισμό τους.

5. Object: είναι μια μη ταξινομημένη συλλογή από ζευγάρια ονομάτων/τιμών όπου τα ονόματα είναι οι strings. Δεδομένου ότι τα objects προορίζονται για την αναπαράσταση προτείνεται, χωρίς να είναι απαραίτητο, κάθε αντικείμενο να έχει μοναδικό κλειδί. Τα αντικείμενα οριοθετούνται με στρογγυλές αγκύλες ,χρησιμοποιούν κόμματα για να χωρίσουν το κάθε ζευγάρι και άνω κάτω τελεία για να δώσουν τιμές στο κλειδί ή όνομα.
6. Null: μια κενή τιμή, χρησιμοποιώντας τη λέξη null(39).

Εικόνα 2-6: Παράδειγμα χρήσης JSON αρχείου

2.5.7. Πρόσβαση στο WinRT με JavaScript

Η JavaScript έχει πρόσβαση στο WinRT μέσω του Windows Namespace. Τα namespaces του δίνουν άμεση πρόσβαση στο περιβάλλον εκτέλεσης της εφαρμογής και την δυνατότητα χρήσης δεδομένων και λειτουργιών της συσκευής όπως πρόσβαση στα δεδομένα, γραφικά, εύρεση τοποθεσίας, κ.τ.λ. (Windows.ApplicationModel, Windows.Data, Windows.Devices, Windows.Graphics). Η διαδικασία δημιουργίας και χρήσης των windows namespaces έγινε πολύ εύκολη από το λειτουργικό αφού κατασκευάστηκε μια νέα Windows Runtime βιβλιοθήκη για JavaScript και ονομάστηκε WinJS. Η WinJS παρέχει (47):

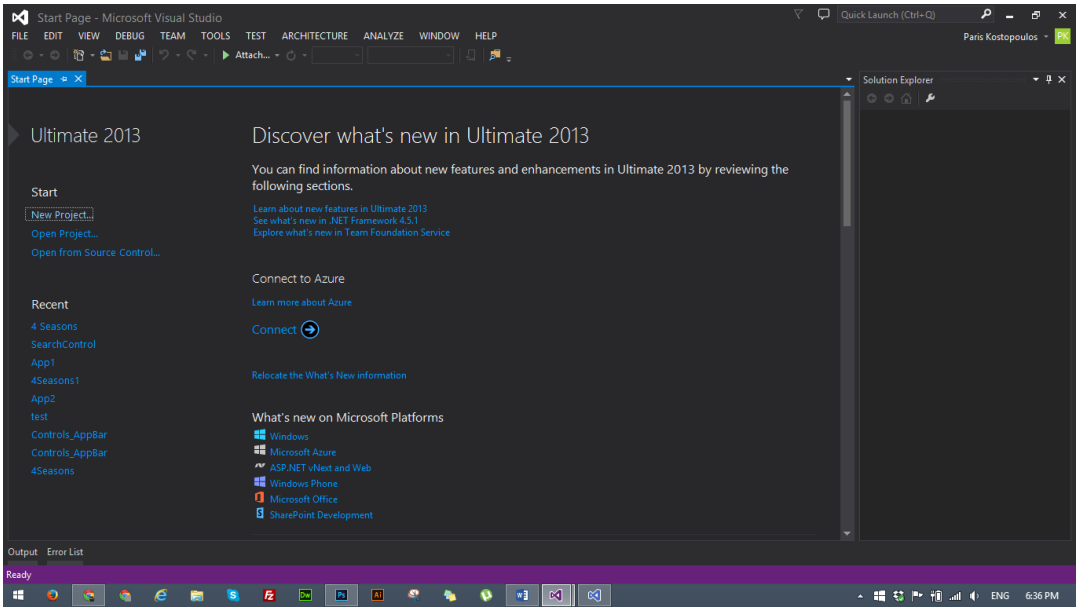
- Πρόσβαση σε αντικείμενα της βιβλιοθήκης (WinJS Objects)
- Μερικά παρόμοια controls με αυτά της Windows.XAML
- Έυκολη πρόσβαση στο App Container (WinJS.Application)
- Απλοποίηση των καταστάσεων προβολής της εφαρμογής
- Πιο εύκολο τρόπο δημιουργίας Namespaces και κλάσεων
- Προσχεδιασμένα στυλ της εφαρμογής (σκοτεινό και φωτεινό θέμα)
- Πρόσβαση σε άλλες βιβλιοθήκες όπως JQuery, κ.α.

2.6. Εργαλεία ανάπτυξης της εφαρμογής

2.6.1. Microsoft Visual Studio

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών της Microsoft. Χρησιμοποιείται για την ανάπτυξη εφαρμογών που εκτελούνται σε Windows OS, ιστοσελίδες, εφαρμογές διαδικτύου και με τις καινούργιες εκδόσεις δίνεται η δυνατότητα ανάπτυξης εφαρμογών για επιπλέον λειτουργικά. Όπως στα περισσότερα προγράμματα ανάπτυξης εφαρμογών, έτσι και εδώ περιλαμβάνεται ένας επεξεργαστής για κώδικα (code editor) καθώς και ένα πρόγραμμα για τον εντοπισμό σφαλμάτων της εφαρμογής και για την διόρθωσή του πηγαίου κώδικα. Επιπρόσθετα το πρόγραμμα μπορεί να δεχτεί και κάποια εξωτερικά plugin που του προσφέρουν επιπλέον δυνατότητες για την υλοποίηση του application.

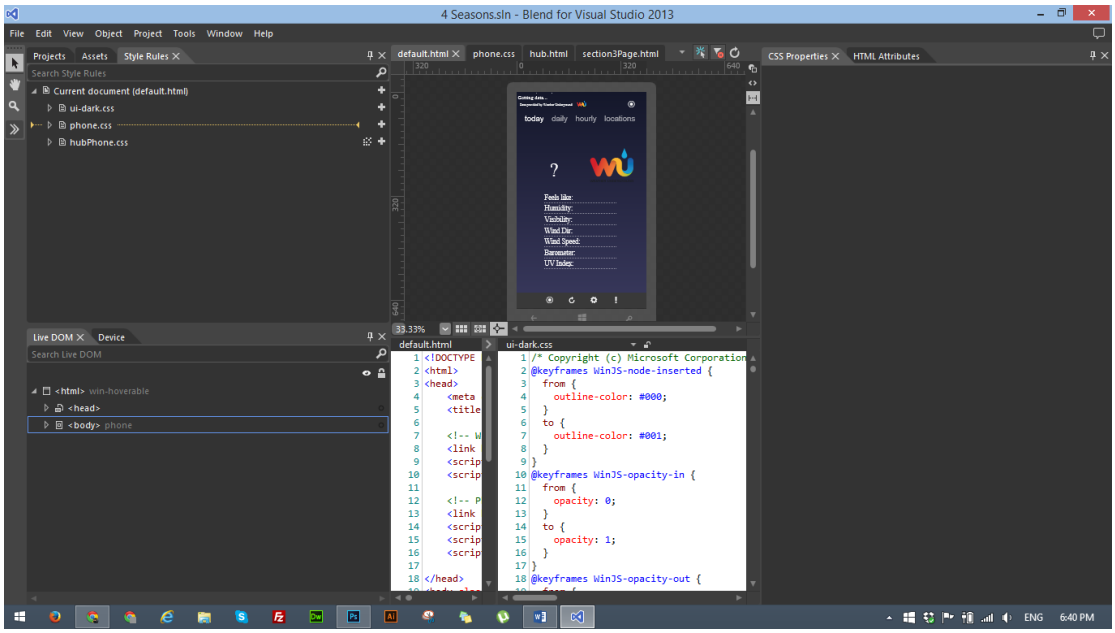
Το Visual Studio έχει σχεδιαστεί να υποστηρίζει πολλές διαφορετικές γλώσσες προγραμματισμού όπως C / C++ και C++/CLI, VB.NET, C#, F# και υποστηρίζει κάποιες άλλες γλώσσες όπως M, Python και Ruby εγκαθιστώντας τις ξεχωριστά. Επιπλέον υποστηρίζει XML / XSLT, HTML / XHTML, JavaScript και CSS. Στις 12 Νοεμβρίου του 2014 η Microsoft ανακοίνωσε μία νέα δωρεάν έκδοση, την Visual Studio Community. Η έκδοση αυτή είναι δωρεάν, υποστηρίζει πολλές γλώσσες (multiple languages), έχει την δυνατότητα χρήσης πρόσθετων εργαλείων και χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής. Η ανάπτυξη εφαρμογής σε αυτό το λογισμικό απαιτεί την εγγραφή του προγραμματιστή ως developer και αυτό μπορεί να γίνει ακόμα και με δωρεάν άδεια έχοντας μερικά δικαιώματα και προνόμια(40).



Εικόνα 2-7: Στιγμιότυπο από το λογισμικό Microsoft Visual Studio

2.6.2. Microsoft Blend for Visual Studio

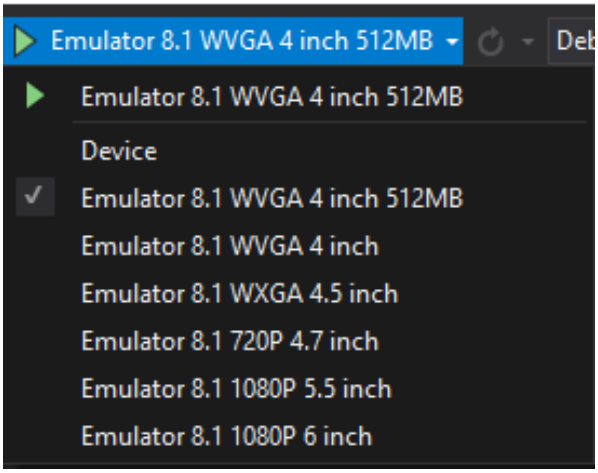
Το Microsoft Blend είναι ένα εργαλείο σχεδιασμού web και desktop εφαρμογών που αναπτύχθηκε από τη Microsoft και στην ουσία συνδυάζει τις δυνατότητες αυτών των δύο είδη εφαρμογών. Στις 24 Ιανουαρίου του 2007, η Microsoft κυκλοφόρησε το πρώτο demo όπου το διένεμε δωρεάν. Ο προγραμματιστής με αυτό το λογισμικό μπορεί να σχεδιάσει τα γραφιστικά στοιχεία που τον ενδιαφέρουν και να τα ενσωματώσει στην εφαρμογή του μέσω του Visual Studio. Στις 17 Οκτώβρη του 2013 κυκλοφόρησε το Blend for Visual Studio 2013 και η εγκατάσταση του γίνεται μαζί με το Visual Studio αν και παράχθηκε ξεχωριστά(41).



Εικόνα 2-8: Στιγμιότυπο από το λογισμικό σχεδιασμού Microsoft Blend

2.6.3. Δοκιμή στην εικονική συσκευή

Στο περιβάλλον που αναπτύσσεται η εφαρμογή, παρέχεται στον προγραμματιστή μια εικονική συσκευή που μπορεί να δοκιμάσει τον κώδικα που έχει γράψει, να δει τα αποτελέσματα και πως αυτά εμφανίζονται κατά την διάρκεια ανάπτυξής της. Αυτό πραγματοποιείται εφόσον ο προγραμματιστής πατήσει το κουμπί “Debug” για να φορτώσει την εφαρμογή στην εικονική συσκευή και για να γίνουν οι απαραίτητες διορθώσεις σε περίπτωση εμφάνισης σφαλμάτων (debugging).



Εικόνα 2-9: Μενού αποσφαλμάτωσης (Debug menu)

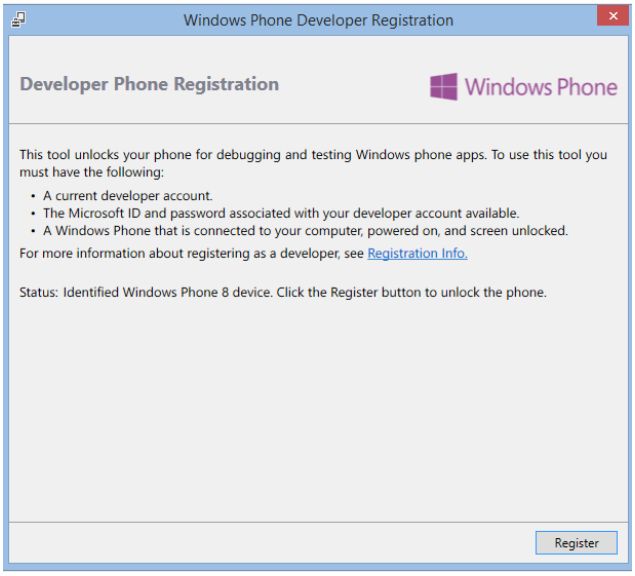
Έτσι λοιπόν οι δυνατότητες της εικονικής συσκευής είναι να εκτελέσει την εφαρμογή στην συσκευή smartphone που τον ενδιαφέρει, να εξερευνήσει την εφαρμογή ακόμα και με «αφή», να πλοηγηθεί στο λειτουργικό του εικονικού κινητού, να περιστρέψει τη συσκευή καθώς και πολλές άλλες χρήσιμες λειτουργίες.

Εικόνα 2-10: Στιγμιότυπα χρήσης της εικονικής συσκευής

2.6.4. Δοκιμή σε πραγματική συσκευή

Στο Visual Studio δίνεται επίσης η δυνατότητα να τεστάρεις και να κάνεις την διαδικασία αποσφαλμάτωσης με πραγματική συσκευή διαλέγοντας στο μενού του Debug το Device. Η διαδικασία που απαιτείται για να υλοποιηθεί αυτό βασίζεται στην

εγκατάσταση του Visual Studio, που σου προφέρει την εφαρμογή Developer Phone Registration.



Εικόνα 2-11: Εγγραφή του κινητού τηλεφώνου ως Developer Phone

ΚΕΦΑΛΑΙΟ 3. ΑΝΑΠΤΥΞΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ «4Seasons»

3.1. Ανάλυση εφαρμογής

3.1.1. Περιγραφή της εφαρμογής

Ρίχνοντας μια ματιά στις εφαρμογές έξυπνων κινητών τηλεφώνων που κυκλοφορούν στα αντίστοιχα καταστήματα εφαρμογών θα διαπιστώσουμε πως τα είδη εφαρμογών έχουν αυξηθεί αρκετά τα τελευταία χρόνια όπως και ο συνολικός αριθμός των εφαρμογών. Στην περίπτωση των Windows Store ο συνολικός αριθμός των εφαρμογών είναι σχετικά μικρός εξαιτίας της μικρής χρονικής περιόδου που έχει κατασκευαστεί σε σχέση με τα άλλα καταστήματα εφαρμογών. Έτσι με βάση τις καθημερινές ανάγκες των καταναλωτών υλοποιήσαμε μια Windows Store εφαρμογή συμβατή με το λειτουργικό Windows Phone 8.1 και παρέχει τα εξής:

- Ενημέρωση καιρικών φαινομένων της παρούσας ημέρας, των επομένων εφτά ημερών καθώς και των επόμενων 24 ωρών μιας τοποθεσίας.
- Εκκίνηση της εφαρμογής με επιλεγμένη τοποθεσία από τον χρήστη, δηλαδή την τοποθεσία που βρίσκεται εκείνη τη στιγμή ή μια προεπιλεγμένη.
- Αναζήτηση και προσθήκη νέων περιοχών σε λίστα με σκοπό την ενημέρωση του καιρού σε τοποθεσίες που ενδιαφέρουν τον χρήστη.
- Απεικόνιση των καιρικών φαινομένων σε διαφορετικές μονάδες μέτρησης όπως την θερμοκρασία σε Celsius (units = metric) ή σε Fahrenheit (units = imperial)

Δημιουργήσαμε την εφαρμογή 4Seasons λόγω ανάγκης για ενημέρωση και συγκεκριμένα του καιρού. Υλοποιεί την απεικόνιση καιρικών φαινομένων σε οποιαδήποτε τοποθεσία με προϋπόθεση την σύνδεσή της σε ένα Web Service μέσω κάποιας τεχνολογίας ασύρματης διαδικτύωσης. Στην συνέχεια αυτού του υποκεφαλαίου θα γίνει η ανάλυση της εφαρμογής και στα επόμενα η σχεδίαση και η υλοποίηση καθώς και παραδείγματα χρήσης για την τεκμηρίωση της λειτουργίας της.

3.1.2. Προδιαγραφές εφαρμογής

Για την δημιουργία της εφαρμογής, την ορθή και ευχάριστη λειτουργία της και την υλοποίηση των παραπάνω αναγκών χρειάστηκαν τα εξής:

- **Επιλογή Windows Phone App προτύπου**: Το Visual Studio διαθέτει διάφορα πρότυπα εφαρμογών (app template). Η σωστή επιλογή του κατάλληλου προτύπου για την κάθε εφαρμογή έτσι ώστε να εξυπηρετεί στην περιήγηση και χρηστικότητά της είναι απαραίτητη.
- **Αρχική σελίδα**: Δημιουργία πρώτης σελίδα που θα ανοίγει κατά την ενεργοποίηση της εφαρμογής καθώς για την απεικόνιση των καιρικών φαινομένων της παρούσας ημέρας.
- **Σελίδα εβδομάδας**: Επίσης απαιτείται μια σελίδα για την απεικόνιση του καιρού των επόμενων 7 ημερών.
- **Σελίδα ωρών**: Ακόμα μια σελίδα για την απεικόνιση του καιρού των επόμενων 24 ωρών σε λίστα.
- **Σελίδα αναζήτησης & αποθηκευμένων τοποθεσιών**: Μια σελίδα για την απεικόνιση της γραμμής αναζήτησης και της λίστας αποθηκευμένων τοποθεσιών.
- **Διαδραστική λίστα τοποθεσιών**: Μια λίστα αποθηκευμένων τοποθεσιών με την δυνατότητα προσθήκης/διαγραφής τοποθεσίας, τη προσθήκη ιδιότητας σε μια από αυτές σε τοποθεσία σπίτι (Set Home) καθώς και εμφάνιση του καιρού με την επιλογή της.
- **Γραμμή Αναζήτησης**: Μια γραμμή αναζήτησης νέας τοποθεσίας από το Web Service μέσω ονόματος της πόλης, ταχυδρομικό κώδικα, κ.α. (Η συγκεκριμένη λειτουργία βασίζεται και στις δυνατότητες αναζήτησης που προσφέρει το Web Service)
- **Κουμπί εύρεσης τοποθεσίας**: Κατά την διάρκεια λειτουργίας της εφαρμογής ένα κουμπί θα προστεθεί για την αναζήτηση της τρέχουσας τοποθεσίας (Current Location) και την απεικόνιση του καιρού.
- **Κουμπί εύρεσης καιρού της τοποθεσίας σπίτι**: Μία ακόμα λειτουργία θα εξυπηρετείτε καθώς ο χρήστης θα έχει στην διάθεση του κουμπί με την εύρεση του καιρού της τοποθεσίας που έχει ορίσει ως τοποθεσία σπίτι.
- **Κουμπί ανανέωσης του καιρού:** Κουμπί για την ανανέωση των στοιχείων του καιρού που θα δείχνει η εφαρμογή σε περίπτωση που τα στοιχεία είναι προηγούμενων ωρών ή επαναφορά από κατάσταση αναμονής μετά από ώρες αλλά ακόμα και μετά από μία ανεπιτυχή λειτουργία της.

- **Σελίδα Ρυθμίσεων**: Σελίδα για την αλλαγή ρυθμίσεων με τις επιλογές του χρήστη οι οποίες θα είναι η αλλαγή της μονάδας μέτρησης που εμφανίζονται τα δεδομένα και η επιλογή της τοποθεσίας που θα βρίσκει τον καιρό η εφαρμογή κατά την εκκίνηση.
- **Σελίδα σχετικά με την εφαρμογή**: Για την ενημέρωση του χρήστη όσον αφορά τον κάτοχο της εφαρμογής, το Web Service που χρησιμοποιεί, την έκδοσή της, κ.α.
- **Web Service**: Ένα από τα πιο σημαντικά και απαραίτητα κομμάτια της εφαρμογής είναι το Web Service το οποίο θα ενημερώνει τα στοιχεία απεικόνισης της εφαρμογής για κάθε τοποθεσία και θα διαθέτει αναζήτηση νέων.
- **Αποθήκευση Ρυθμίσεων**: Απαραίτητη προϋπόθεση για την μόνιμη αποθήκευση των ρυθμίσεων είναι η πρόσβαση στα αρχεία της εφαρμογής με την δυνατότητα εγγραφής/διαβάσματος.
- **Πολυγλωσσική εφαρμογή:** Το πρότυπο που θα βασιστούμε θα πρέπει να έχει την δυνατότητα επέκτασης γλωσσών στο μέλλον.
- **Responsive σχεδιασμός**: Αυτό το είδος σχεδιασμού βασίζεται σε μια νέα τεχνοτροπία που έχει γίνει γνωστή από τις ιστοσελίδες. Είναι μια προσέγγιση δημιουργίας ιστοσελίδων που θα μπορούν να ανταποκριθούν τα αντικείμενα προς απεικόνιση σε διαφορετικά μεγέθη και αναλύσεις οθόνης,. Αυτό το στυλ μπορεί να χρησιμοποιηθεί και για τον σχεδιασμό της εφαρμογής έτσι ώστε να ανταποκρίνεται σε όλα τα μεγέθη και αναλύσεις οθόνης των κινητών.
- **Εικονίδια ένδειξης καιρού**: Τα εικονίδια που συνοδεύουν την όλη περιγραφή του καιρού είναι απαραίτητα καθώς στις μέρες μας τα smartphone δεν έχουν καμία δυσκολία απεικόνισης δεδομένων τέτοιου τύπου όπως εικόνας, animation κ.α.
- **Άλλα γραφικά**: Άλλα γραφιστικά πολυμέσα θα ήταν καλό να υπάρχουν για την καλή θέση της εφαρμογής στην αγορά, την αναγνώρισή της από ένα λογότυπο, την δημοτικότητά της κ.α.

3.1.3. Συμβατότητα εφαρμογής

Όλες οι εφαρμογές σχεδιάζονται και κατασκευάζονται ώστε να παρέχουν την καλύτερη δυνατή εμπειρία στους σύγχρονους χρήστες. Ένα μεγάλο ποσοστό εφαρμογών για smartphones είναι διαθέσιμα σχεδόν σε όλα τα λειτουργικά συστήματα

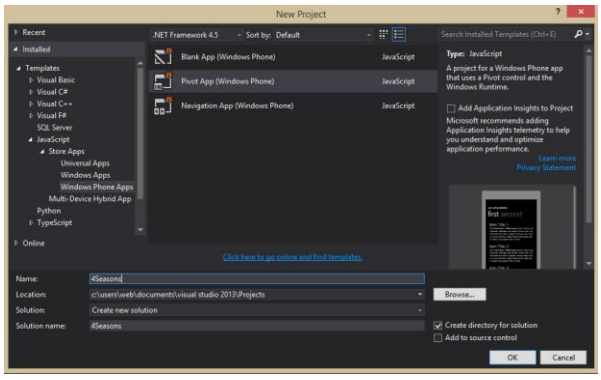
της αγοράς. Ωστόσο το πρόβλημα συμβατότητας έγκειται στη διαφοροποίηση και τα τεχνικά χαρακτηριστικά που διαθέτει κάθε συσκευή.

Η εφαρμογή 4Seasons υποστηρίζεται από όλα τα smartphones της σειράς Lumia με ελάχιστη διαγώνιο οθόνης 3,7 ίντσες, μνήμη RAM 512 MB και επεξεργαστή 1.4GHz όπως το Lumia 800 και το Lumia 710 όπου ήταν και οι ναυαρχίδες της σειράς Lumia της Nokia. Τα επόμενα μοντέλα όπως το Lumia 1320 με διαγώνιο οθόνης 6 ίντσες και το Lumia 640 XL με διαγώνιο 5,7 ίντσες και σαφώς καλύτερα χαρακτηριστικά από τα προηγούμενα μοντέλα δίνουν μια πιο ευχάριστη περιήγηση στην εφαρμογή. Ωστόσο η εφαρμογή σχεδιάστηκε να αναπτύχθηκε με γνώμονα το Lumia 635 με διαγώνιο 4,5 ίντσες, μνήμη RAM 512 MB και επεξεργαστή 1.2GHz. Αυτό πρακτικά σημαίνει ότι η απεικόνιση της εφαρμογής καλύπτει πλήρως το εύρος θέασης της οθόνης της συγκεκριμένης συσκευής.

3.2. Σχεδίαση εφαρμογής

3.2.1. Εισαγωγή

Μετά από την ανάλυση και καταγραφή των προδιαγραφών μιας εφαρμογής με στόχο την απεικόνιση του καιρού, όπου πολλές από αυτές είναι γνωστές, θα ήταν καλό ο σχεδιασμός της να βασιστεί εξαρχής στον καλύτερο συνδυασμό των παραπάνω και να χαρακτηρίζεται από αξιοπιστία, ασφάλεια, ορθότητα, ευχρηστία και άλλα πολλά τέτοια θετικά χαρακτηριστικά μιας εφαρμογής. Ένα μεγάλο μέρος σε αυτό τον στόχο συμβάλει η κατανόηση του λειτουργικού συστήματος που πρόκειται να είναι συμβατή η εφαρμογή. Όπως προ είπαμε στις προδιαγραφές, το Visual Studio παρέχει πρότυπα που μπορεί να βασιστεί ο προγραμματισμός της εφαρμογής. Το πρότυπο που επιλέχθηκε για την υλοποίηση της εφαρμογής είναι το Pivot App Template από τα διαθέσιμα, για Windows Phone λειτουργικό και γλώσσα προγραμματισμού την JavaScript.



Εικόνα 3-1: Pivot Template για Windows Phone Apps στο Visual Studio

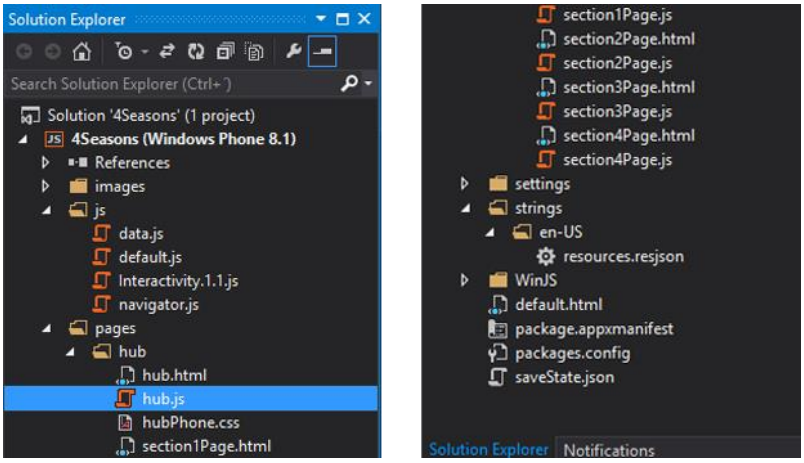
3.2.2. Pivot App Template για Windows Phone

Το πρότυπο αυτό προσφέρει έναν γρήγορο τρόπο διαχείρισης και παρουσίασης δεδομένων αφού χρησιμοποιείται κυρίως για το φιλτράρισμα και την απεικόνιση δεδομένων μεγάλου μεγέθους. Η περιήγηση στο συγκεκριμένο πρότυπο βασίζεται σε έναν μηχανισμό που σύρεις τις σελίδες αριστερά-δεξιά (Carousel) και η εμφάνιση των σελίδων είναι ιεραρχική. Αποτελείται από σελίδες HTML σε συνεργασία με αρχεία CSS για την απαραίτητη μορφοποίηση και αρχεία JavaScript για τα δυναμικά αποτελέσματα της εφαρμογής με την προσθήκη συνδέσμου των αρχείων στην ετικέτα **<head>**. Ο κώδικας JavaScript για μία σελίδα δεν είναι απαραίτητο να βρίσκεται σε δικό του ξεχωριστό αρχείο, αλλά μπορεί να γραφεί σε ένα αρχείο .js που εκφράζει πολλές σελίδες(42)(43).

Από την πλευρά της γλώσσας προγραμματισμού η σύνδεση με την σελίδα γίνεται χρησιμοποιώντας αντικείμενο της WinJS βιβλιοθήκης που αφορά τον προσδιορισμό των σελίδων **WinJS.UI.Pages.define (uri, members);**. Σε αυτό το αντικείμενο τα στοιχεία της παρένθεσης είναι ο σύνδεσμος του μονοπατιού της σελίδας μέσα στην εφαρμογή (**uri**) και οι διάφορες καταστάσεις που μπορεί να βρεθεί η σελίδα κατά την διάρκεια χρήσης της ενώ περιλαμβάνει και τις λειτουργίες προς εκτέλεση σε αυτές τις καταστάσεις (**members**). Κάποια παραδείγματα τιμών που μπορεί να πάρει στοιχείο **member** είναι 1) **processed**: εκτέλεση εντολών πριν την αρχικοποίηση της σελίδας και την κοινοποίηση στοιχείων της, 2) **ready**: εκτέλεση εντολών και χειρισμός συναρτήσεων από κουμπιά και άλλα τέτοια αντικείμενα αφού η σελίδα είναι έτοιμη και έχει κοινοποιήσει το περιεχόμενο της 3) **unload**: εκτέλεση εντολών κατά την διάρκεια απομάκρυνσης από την σελίδα πηγαίνοντας σε μία άλλη(44)(45).

```
WinJS.UI.Pages.define ("/pages/hub/hub.html", {
    processed: function (element) {
        return WinJS.Resources.processAll(element);
    },
    // This function is called whenever a user navigates to this page. It
    // populates the page elements with the app's data.
    ready: function (element, options) {
        var hub = element.querySelector(".hub").winControl;
        hub.onheaderinvoked = function (args) {
            args.detail.section.onheaderinvoked(args);
        }
        // TODO: Initialize the page here.
    },
    unload: function () {
        // TODO: Respond to navigations away from this page.
    },
});
```

Στο πρότυπο Pivot η ακολουθία διαβάσματος των σελίδων είναι προγραμματισμένες. Κατά την ενεργοποίηση της εφαρμογής εκτελείται πρώτα η **default.html** σελίδα καθώς και η αντίστοιχη JavaScript όπου περιέχει σημαντικές εντολές για την λειτουργία της εφαρμογής και την επαναφορά της μετά από κατάσταση αναμονής. Επίσης η σελίδα **default.html** εμπεριέχει την σύνδεση της εφαρμογής με τα απαραίτητα αρχεία .js για την περιήγηση και τα δεδομένα τοποθεσίας (**navigator.js – data.js**). Ως μοναδικό περιεχόμενο προς κοινοποίηση είναι η εμφωλευμένη σελίδα **hub.html** που βρίσκεται ένα επίπεδο πιο μέσα στο δένδρο των αρχείων, συγκεκριμένα στον φάκελο pages. Το σημείο hub στην εφαρμογή είναι ο κόμβος που ενώνονται όλα μεταξύ τους. Ακολουθείται από ένα **hubphone.css** αρχείο, έχοντας την απαραίτητη μορφοποίηση για το διαφορετικό πρότυπο που διαλέξαμε και ένα **hub.js** αρχείο που είναι υπεύθυνο σχεδόν για όλες τις λειτουργίες της εφαρμογής. Η **hub.html** έχει ως περιεχόμενο άλλες 4 σελίδες (sections) προς κοινοποίηση όπου σε αυτές θα βασιστούμε για τον διαχωρισμό της απεικόνισης των δεδομένων. Ο σχεδιασμός θα είναι αλλαγές σε αυτές τις 4 προτυποποιημένες σελίδες αν και υπάρχει η δυνατότητα διαγραφής τους και δημιουργίας νέων(42)(43).

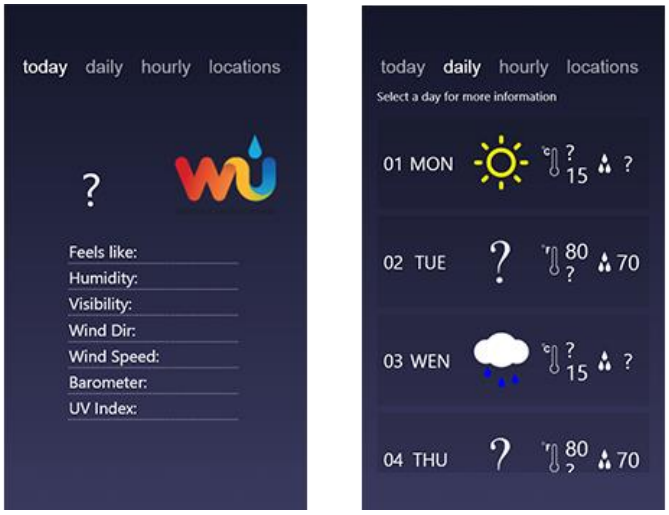


Εικόνα 3-2: Δένδρο αρχείων μιας Pivot εφαρμογής

3.2.3. Σχεδιασμός σελίδων

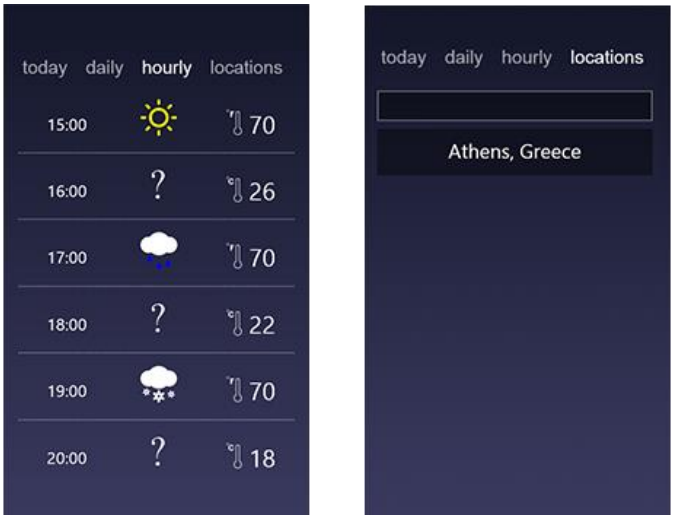
Η εφαρμογή θα αποτελείται από την πρώτη σελίδα του κόμβου προς εμφάνιση, την **section1Page.html**, με τίτλο **today** και στοιχεία προς απεικόνιση τα καιρικά φαινόμενα της παρούσας ημέρας. Πιο αναλυτικά θα υπάρχουν οι ενδείξεις για την πλήρη ονομασία της τοποθεσίας, η περιγραφή του καιρού, η θερμοκρασία, το εικονίδιο ένδειξης του καιρού και ένας πίνακας με την απεικόνιση της θερμοκρασίας feels like, την υγρασία, την ορατότητα, την ταχύτητα και την κατεύθυνση του ανέμου, το

βαρόμετρο και τέλος τον δείκτη UV. Η δεύτερη σελίδα, **section2Page.html** θα έχει τίτλο **daily** και περιεχόμενο μια λίστα καιρού των επόμενων 7 επόμενων ημερών και πιο συγκεκριμένα θα απεικονίζει τις ενδείξεις ημερομηνίας, ημέρας, εικονίδιο του καιρού, μεγαλύτερη/μικρότερη θερμοκρασία και υγρασία.



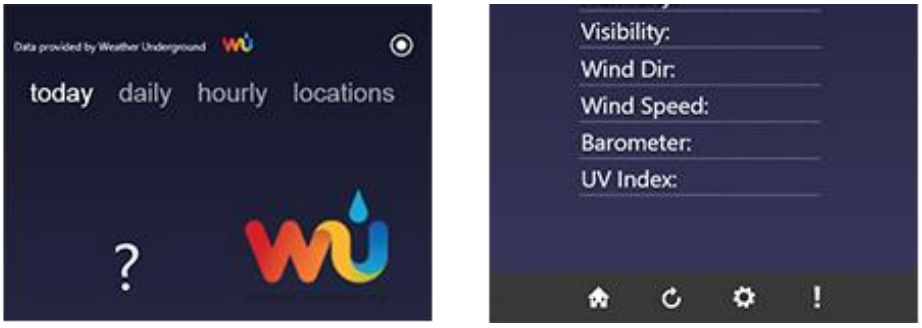
Εικόνα 3-3: Στιγμιότυπα σχεδιασμού σελίδων today – daily

Η τρίτη σελίδα **section3Page.html** με όνομα **hourly** θα έχει προς απεικόνιση τα στοιχεία ώρας (Μορφή Ώρας - 24ωρο), το εικονίδιο του καιρού και την θερμοκρασία των επόμενων 24 ωρών. Τέλος η τέταρτη σελίδα **section4Page.html** που θα ονομαστεί **locations** θα διαθέτει μία γραμμή αναζήτησης για την εύρεση τοποθεσίας από το Web Service και επιλέγοντας κάποιο από τα αποτελέσματα θα γίνεται αποθήκευση στην λίστα. Στον σχεδιασμό αυτής της διαδραστική λίστας πρέπει να λάβουμε υπόψιν πως θα υπάρχει πάντα μία τοποθεσία τουλάχιστον και θα έχει την ιδιότητα τοποθεσίας.



Εικόνα 3-4: Στιγμιότυπα σχεδιασμού σελίδων hourly – locations

Υπάρχουν και άλλες σελίδες προς σχεδίαση που δεν θα κοινοποιούνται με την εκκίνηση της εφαρμογής αλλά μόνο όταν ο χρήστης περιηγηθεί προς αυτές. Χωρίς να είναι εξαρτώμενες σελίδες από άλλες, όπως η hub.html από την default.html, θα σχεδιάσουμε μία σελίδα ρυθμίσεων **settings.html** συνδεδεμένη με έναν αρχείο μορφοποίησης **settings.css** για τα επιπλέον στοιχεία html που θα περιέχει. Στην συνέχεια παρόμοιο σχεδιασμό θα έχει η σελίδα ενημέρωσης σχετικά με την εφαρμογή που θέλουμε να σχεδιάσουμε (**about.html).**. Περιεχόμενο της σελίδας ρυθμίσεων θα είναι ένα πλαίσιο γενικών ρυθμίσεων με την επιλογή μιας από τις δύο (Radio Buttons) μονάδες μέτρησης για την απεικόνιση των στοιχείων καιρού ανάμεσα σε metric και imperial (π.χ. η θερμοκρασία σε Celsius/Fahrenheit, η απόσταση σε km/mi, η ταχύτητα σε Kph/Mph, κ.τ.λ.) και μιας από τις δύο τοποθεσίες εκκίνησης (Current Location/Home Location). Για να μπορεί ο χρήστης να περιηγηθεί σε αυτές τις σελίδες θα προστεθούν κουμπιά μεταφοράς σε αυτές, αλλά και κουμπιά άλλων χρήσεων σύμφωνα με τις προδιαγραφές της εφαρμογής. Τα πρώτα 4 κουμπιά που θα εισαχθούν θα είναι στην κάτω μπάρα της οθόνης (**appBar**) για την εύρεση του καιρού της τοποθεσίας σπίτι (**Home Location Button**), για την ανανέωση του καιρού της τοποθεσίας (**Refresh Button**) και άλλα δύο κουμπιά για την μετάβαση του χρήστη στις σελίδες **settings** και **about**. Το αντικείμενο της βιβλιοθήκης WinJS που λέγεται appBar (κάτω μπάρα) θα βρίσκεται στην **default.html** και θα παύει να εμφανίζεται όταν απομακρύνεται από αυτήν, για παράδειγμα θα οδηγείτε στην σελίδα ρυθμίσεων έτσι ώστε η μόνη δυνατότητα περιήγησης από τις άλλες σελίδες να είναι η επιστροφή με το back button της συσκευής. Ακόμα ένα κουμπί στο πάνω μέρος της οθόνης θα προστεθεί για την εύρεση τοποθεσίας του χρήστη και του καιρού αυτής (**Current Location Button**). Το κουμπί εύρεσης τοποθεσίας θα βρίσκεται επίσης στην σελίδα **hub.html** για να μπορεί ο χρήστης να το πατήσει ενώ βρίσκεται σε οποιαδήποτε από τις 4 σελίδες απεικόνισης του καιρού αλλά όχι όταν απομακρυνθεί σε άλλη σελίδα.



Εικόνα 3-5:Σχεδιασμός κουμπιών έυρεσης τοποθεσίας & κάτω μπάρας

3.2.4. Weather Underground Web Service

Η διαδικτυακή υπηρεσία που θα χρησιμοποιεί η εφαρμογή για να έχει στην διάθεση της τα καιρικά φαινόμενα είναι και αυτή που θα μπορεί να την κάνει γρήγορη και αξιόπιστη κατά μεγάλο βαθμό. Τα Web Services που μεταδίδουν τον καιρό έχουν εξελιχθεί αρκετά τα τελευταία χρόνια και η αναζήτηση ενός που θα κάνει οικονομία στο μέσο διαδικτύωσης, θα έχει μικρό χρόνο απόκρισης από την πλευρά του server και παράλληλα καλή σύνταξη των δεδομένων για την εύκολη ανάγνωση από τον προγραμματιστή, προσφορά ποικιλίας δεδομένων όπως απεικόνιση σε πολλές γλώσσες ή σε διάφορες μονάδες μέτρησης θα ήταν ότι καλύτερο. Μετά από αναζήτηση στο διαδίκτυο το Web Service για τα δεδομένα του καιρού που επιλέχθηκε είναι της εταιρίας Weather Underground(http://www.wunderground.com/).

Τα προσφερόμενα αντικείμενα σύμφωνα με τις απαιτήσεις μας από το Web Service είναι η μεγάλη γκάμα δεδομένων για την δυνατότητα απεικόνισης των στοιχείων ανά σελίδα που έχουμε σχεδιάσει παραπάνω αλλά και άλλων που θα μπορούσαν να συμβάλουν στην έξυπνη συμπεριφορά της εφαρμογής (π.χ. όταν η εφαρμογή θα απεικονίζει τον καιρό και είναι μέρα, θα έχει φωτεινή ταπετσαρία ενώ αν είναι βράδυ, θα έχει σκοτεινή). Επίσης τα δεδομένα δίνονται στο τύπο αρχείων JSON που έχουμε επιλέξει μιας και ο κύριος λόγος ύπαρξής τους είναι για την μεταφορά δεδομένων μεταξύ Server/Client. Παρέχει επίσης την καλή σύνταξη της γλώσσας JSON έχοντας στα αντικείμενα σχετικό τίτλο (π.χ. το αντικείμενο θερμοκρασία σε Celsius αναγράφεται με τον τίτλο temp_c), συνεργάζεται με πολλούς μετεωρολογικούς σταθμούς ανά γεωγραφικές περιοχές προσθέτοντας αξιοπιστία στα δεδομένα της και προσφέρει μία κλήση για όλα τα δεδομένα (της παρούσας ημέρας, των επόμενων ημερών και ωρών) δίνοντας στην εφαρμογή το πλεονέκτημα της ταχύτητας και της οικονομίας κίνησης δεδομένων.

Η κλήση στο Web Service γίνεται με έναν σύνδεσμο που παρέχει το API της ιστοσελίδας. Επίσης απαιτείται να εμπεριέχεται στον σύνδεσμο ένα κλειδί διεπαφής προγραμματισμού της εφαρμογής (API key) για να αναγνωρίζει η ιστοσελίδα το πρόγραμμα-λειτουργία που καλείται και τον προγραμματιστή ή τον χρήστη που καλεί. Ακόμα χρησιμοποιείτε για να εντοπίζει και να χειρίζεται τον τρόπο που θα χρησιμοποιηθεί το API, για παράδειγμα να αποφευχθεί κάθε είδους κακόβουλη χρήση ή κατάχρηση (όπως ορίζεται ίσως από τους όρους της υπηρεσίας).

Εικόνα 3-6: Λογότυπο του Weather Underground (57).

3.2.5. Κύκλος ζωής της εφαρμογής

Σχεδιάζοντας τον κύκλο ζωής της εφαρμογής μπορούν να κατανοηθούν καλύτερα οι περιπτώσεις χρήσης καθώς και ο τρόπος που θα διαχειρίζεται η εφαρμογή του πόρους της συσκευής. Οι περιπτώσεις εκκίνησης της εφαρμογής είναι στην πρώτη περίπτωση νέα εκκίνηση ή μετά από τερματισμό και στην άλλη μετά από προσωρινή αποθήκευση και κατάσταση αναμονής. Μια εφαρμογή ξεκινά με την συμπεριφορά πρώτης εκκίνησης μόλις ενεργοποιηθεί από τις καταστάσεις notRunning, closedByUser ή terminated. Στην περίπτωση αυτή η εφαρμογή θα εκτελεί εντολές όσον αφορά την ενεργοποίηση του αντικειμένου app και την προετοιμασία στοιχείων και ελέγχου για τις υπόλοιπες σελίδες, έχοντας ολοκληρώσει τις καταστάσεις σελίδων processed και ready. Θα διαβάζει τα τελευταία δεδομένα και στοιχεία από την μόνιμη αποθήκευση και θα ελέγχει τις προτιμήσεις του για τοποθεσία εκκίνησης. Αν η επιλογή του είναι να βρει τον καιρό της τρέχουσας τοποθεσίας του θα πρέπει να εντοπίσει την θέση της συσκευής, να κατασκευάσει έναν σύνδεσμο με τις συντεταγμένες και να γίνει κλήση στο Web Service για να μας επιστρέψει τα δεδομένα προς απεικόνιση. Σε άλλη περίπτωση θα βρεθεί ο καιρός για την τοποθεσία σπίτι με έναν κατασκευασμένο σύνδεσμο εξαρχής και θα εκτελεστεί η επόμενη συνάρτηση αυτή της κλήσης.

Αφού μεταβεί στην επόμενη κατάσταση running έχοντας ολοκληρώσει την ενεργοποίηση οι συναρτήσεις αλληλοεπιδρούν με τις επόμενες έτσι ώστε η μία να πυροδοτεί την άλλη και να φτάνει στο επιθυμητό αποτέλεσμα. Μετά από την κατασκευή του συνδέσμου και μεταβαίνοντας στην εκτέλεση της συνάρτησης κλήσης προς το Web Service θα πρέπει η εφαρμογή να μπορεί να διαχειριστεί την κάθε περίπτωση επιτυχίας ή αποτυχίας της κλήσης. Σε περίπτωση αποτυχίας της λήψη δεδομένων ή όταν ο χρήστης δεν έχει πρόσβαση στο internet, ακόμα και στην πρώτη εκκίνηση που σημαίνει πως η εφαρμογή δεν θα έχει δεδομένα προς απεικόνιση, με τις απαραίτητες εντολές θα μεταβαίνει στο διάβασμα των δεδομένων της τελευταίας κλήσης και στην απεικόνισή τους. Αν η κλήση είναι επιτυχής, θα αποθηκεύει τα δεδομένα και θα συνεχίζει στο γέμισμα των πεδίων.

Μια επόμενη συνάρτηση είναι αυτή που θα υλοποιεί την κοινοποίηση των στοιχείων στις αντίστοιχες σελίδες. Έχοντας σε κάθε περίπτωση δεδομένα προς

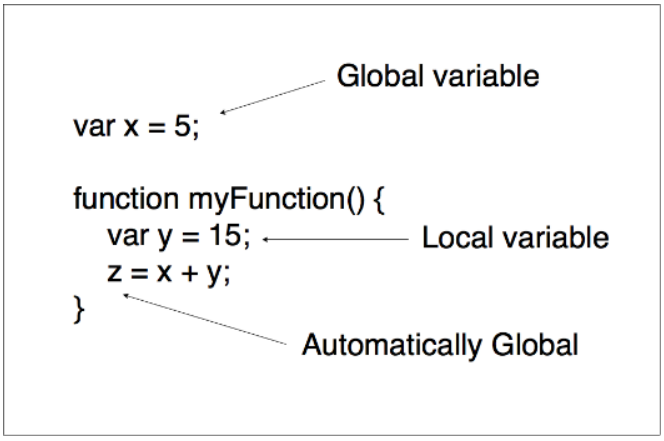
απεικόνιση, θα παίρνει τα απαραίτητα στοιχεία και θα τα τοποθετηθεί στα κελία που είναι προγραμματισμένη ανά σελίδα. Αυτή η συνάρτηση θα είναι επίσης βασικό κομμάτι λειτουργίας της εφαρμογής γιατί κάθε φορά που ο χρήστης απομακρύνεται από την κομβική σελίδα που έχει ως περιεχόμενα τις 4 ενότητες τα κελιά αδειάζουν και παύουν να έχουν τιμές, οπότε κατά την επιστροφή του σε αυτές θα πρέπει να αναλαμβάνει το γέμισμα των πεδίων ξανά. Επιπλέον αν ο χρήστης μεταβεί στην σελίδα ρυθμίσεων και αλλάξει τις μονάδες μέτρησης θα πρέπει στην επιστροφή να γεμίσουν τα πεδία με τα αντίστοιχα νέα στοιχεία χωρίς νέα κλήση. Η αποφυγή κλήσης και χρέωσης δεδομένων είναι αναγκαίο για την ποιοτική και έξυπνη συμπεριφορά της εφαρμογής.

Από την κατάσταση running μια περίπτωση είναι να τεθεί σε κατάσταση αναμονής. Εκτελώντας ο χρήστης εναλλαγή εφαρμογών και δίνοντας στην εφαρμογή το χρονικό περιθώριο για checkpoint θα αποθηκεύεται η ώρα τελευταίας ανανέωσης των δεδομένων και η τιμή μιας μεταβλητής activated έτσι ώστε αν μεταβεί ξανά σε λιγότερο από 3 ώρες να επαναφέρει τα δεδομένα που είχε προς απεικόνιση διαφορετικά θα ξεκινήσει από την αρχή αναζητώντας νέα δεδομένα. Τέλος, η εφαρμογή σε όλες τις πιθανές περιπτώσεις απενεργοποίησης έχει πάντα αποθηκευμένα δεδομένα προς απεικόνιση ακόμα και από την πρώτη ενεργοποίηση μετά από την εγκατάστασή της.

3.3. Υλοποίηση εφαρμογής

3.3.1. Μεταβλητές

Σύμφωνα με την σχεδίαση και την ροή της εφαρμογής σε αυτό το υποκεφάλαιο θα γίνει η περιγραφή συναρτήσεων που κατασκευάστηκαν και τα αντικείμενα της WinJS βιβλιοθήκης που χρησιμοποιήθηκαν για την υλοποίηση. Η λειτουργία της εφαρμογής βασίζεται κυρίως στο αρχείο **hub.js**. Προσδιορίζοντας σε ένα αρχείο τον προγραμματισμό των σελίδων έχει ως αποτέλεσμα την έυκολη και γρήγορη εκτέλεση των συναρτήσεων αλλά και την χρήση μεταβλητών από όλες τις συναρτήσεις. Στην γλώσσα προγραμματισμού JavaScript όταν μια μεταβλητή δηλώνεται έξω από τις συναρτήσεις είναι ορατή από όλες (**global**), όταν τις δηλώσουμε μέσα στην συνάρτηση θα είναι ορατή μόνο από την ίδια την συνάρτηση (**local**) και αν γραφεί σε συνάρτηση χωρίς να δηλωθεί γίνεται αυτόματα ορατή από όλες τις συναρτήσεις (**Automatically Global**).



Εικόνα 3-7: JavaScript Scope for variables

Οι μεταβλητές που δηλώθηκαν έξω από τις συναρτήσεις ήταν αρκετές μιας και αυτό εξυπηρετούσε στην όλη λειτουργία της εφαρμογής. Για παράδειγμα η μεταβλητή **activated** καθορίζει κατά την εκκίνηση της εφαρμογής αν είναι η πρώτη ενεργοποίηση ή αν επανέρχεται από κατάσταση αναμονής έχοντας την τιμή false ή true αντίστοιχα (First Activation - Suspended), η μεταβλητή **HomeLocationURL** είχε ως τιμή το URL που χρειάζεται η συνάρτηση κλήσης στο web service για να βρει απευθείας τον καιρό της τοποθεσίας που είχε την ιδιότητα σπίτι, η μεταβλητή **unitsVariable** είχε ως τιμή την μονάδας μέτρησης Celsius ή Fahrenheit που θα πρέπει να απεικονίζονται τα στοιχεία, κ.α.(46).

3.3.2. Μόνιμης αποθήκευση ρυθμίσεων (read/write)

Για την αποθήκευση δεδομένων σε μία εφαρμογή υπάρχουν πολλοί τρόποι και συνήθως υλοποιείται με την χρήση αρχείων της εφαρμογής, διαφορετικού τύπου (π.χ. **.json - .txt**) και με την δυνατότητα εγγραφής και διαβάσματος σε αυτό. Στην εφαρμογή 4Seasons το αρχείο που θα κρατάει όλες τα επιθυμητά στοιχεία προς αποθήκευση θα είναι το **saveState.json**. Θα περιέχει τρία αντικείμενα στο πρώτο επίπεδο όπου το πρώτο θα αφορά την καταγραφή των τοποθεσιών με στοιχεία το **όνομα της πόλης,** αν η έχει την **ιδιότητα της τοποθεσίας σπίτι** και το **εικονίδιο** home που θα φέρει το αντικείμενο της λίστας. Το στοιχείο αυτό δεν θα μπορεί να διαγραφεί αν δεν πάρει το αντικείμενο **home** την τιμή **false** με αυτό να σημαίνει πως κάποιο άλλο αντικείμενο θα την έχει πάντα. Το δεύτερο αντικείμενο θα αφορά τις ρυθμίσεις για την **μονάδα μέτρησης** που θα απεικονίζονται τα στοιχεία και την **τοποθεσία εκκίνησης** της εφαρμογής που επιθυμεί ο χρήστης καθώς και το τρίτο στοιχείο θα αποθηκεύει τα τελευταία **δεδομένα** που έκανε λήψη από την τελευταία επιτυχημένη κλήση.

JSON (saveState.json)

```
{
   "LOCATIONS": [
      { "name": "Athens, Greece", "home": true , "img": "/images/homeflag.png" },
      { "name": "Patrai, Greece", "home": true , "img": "/images/emptyflag.png" }
   ],

   "SETTINGS": { "unitsVariable": "Celsius", "launchLocationVariable": "HomeLocation" },

   "DATA":  {
            "response": {
            "version":"0.1",
            "termsofService":"http://www.wunderground.com/weather/api/d/terms.html",
            "features": {
            "conditions": 1
            ,
            "hourly": 1
            ,
            "forecast10day": 1
            ,
            "astronomy": 1
            }
```

Το αρχείο αυτό το διαβάζεται κατά την ενεργοποίηση της εφαρμογής από την σελίδα **data.js** μιας και εκτελείται πρώτη κατά την εκκίνηση. Συγκεκριμένα με ήδη υπάρχων τον σύνδεσμο του αρχείου που έχει αποθηκευτεί χρησιμοποιείται το αντικείμενο **Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url)** για την ανάθεση του αρχείου από τον χώρο αποθήκευσης της συσκευής και διαβάζεται με το αντικείμενο **Windows.Storage.FileIO.readTextAsync(file)**. Στην συνέχεια δημιουργεί την λίστα των τοποθεσιών, μεταφέρει το αρχείο στην σελίδα **hub.js** και ξεκινά η εκτέλεση των συναρτήσεών της με πρώτη την **readSettingsData().** Μετά από έλεγχο της μεταβλητής **activated** για την αναγνώριση της πρώτης εκκίνησης έχοντας το αρχείο προς διάβασμα αρχίζει και μοιράζει σε όλες τις μεταβλήτες που είναι προγραμματισμένο τις αντίστοιχες τιμές, κατασκευάζει τον σύνδεσμο για την τοποθεσία σπίτι, μετατρέπει την μεταβλητή ένδειξης αν έχει ενεργοποιηθεί η εφαρμογή σε true και εκτελεί την επόμενη συνάρτηση.

JAVASCRIPT (hub.js)

```
// LOAD LAST SAVED SETTINGS
   var saveStateData;
   var HomeLocationURL;
   function readSettingsData() {
      unitsVariable = saveStateData.SETTINGS.unitsVariable;
      launchLocationVariable = saveStateData.SETTINGS.launchLocationVariable;
      data = saveStateData.DATA;
      for (var i = 0, len = saveStateData.LOCATIONS.length; i < len; i++) {
         if (saveStateData.LOCATIONS[i].home == true) {
            HomeLocationURL =
"http://api.wunderground.com/api/ee4561125223576a/conditions/hourly/forecast10day/astronomy/q/" +
saveStateData.LOCATIONS[i].name + ".json";
         }
      }
```

```
        activated = true;
        WinJS.Application.addEventListener = startWeather();
    }
```

3.3.3. Συναρτήσεις εύρεσης καιρού

Η συνάρτηση **startWeather()** ελέγχει την μεταβλητή που προσδιορίζει την τοποθεσία εκκίνησης και εκτελεί αντίστοιχα την εύρεση τοποθεσίας ή απευθείας την κλήση στο Web Service. Έχοντας έναν έλεγχο **if** και την συνθήκη, εκτελεί τις επόμενες συναρτήσεις χρησιμοποιώντας το αντικείμενο **WinJS.Application.addEventListener** όπου στην πρώτη περίπτωση είναι η **getLoc()** (εύρεση τοποθεσίας) και στην δεύτερη η **getWeatherJSON(baseURL)** που απαιτεί είσοδο ένα URL προς κλήση (κλήση στο Web Service).

```
JAVASCRIPT (hub.js)

// FUNCTION - This function call weather JSON by the checked launchLocationVariable
    function startWeather() {
        if (launchLocationVariable == "CurrentLocation") {
            WinJS.Application.addEventListener = getLoc();
        } else {
            var baseURL = HomeLocationURL;
            setTimeout(function () {
                WinJS.Application.addEventListener = getWeatherJSON(baseURL);
            }, 1);
        }
    }
```

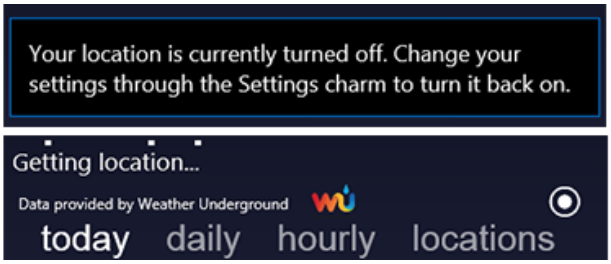
Για να έχει η συνάρτηση, κατά την εκτέλεσή της, πρόσβαση στην συσκευή εύρεσης τοποθεσίας πρέπει η συγκεκριμένη υπηρεσία να είναι ενεργοποιημένη. Στα αρχεία της εφαρμογής περιλαμβάνεται το **package.appxmanifest**. Από αυτό το αρχείο ο προγραμματιστής προσδιορίζει τις λειτουργίες που θα πρέπει να είναι ενεργοποιημένες έτσι ώστε η εφαρμογή να λειτουργήσει ορθά. Έχοντας λοιπόν ενεργοποιήσει την υπηρεσία εύρεσης τοποθεσίας η συνάρτηση **getLoc()** ελέγχει αν υπάρχει κάποια τελευταία τοποθεσία στην μνήμη και στην συνέχεια εκτελεί ένα αίτημα εύρεσης τοποθεσίας με την εντολή πρόσβασης στην αντίστοιχη συσκευή (**Windows.Devices.Geolocation.Geolocator**). Επομένως αν η εύρεση τοποθεσίας είναι επιτυχής καταγράφονται οι συντεταγμένες, κατασκευάζεται το URL προς κλήση στο Web Service και εκτελείτε η επόμενη συνάρτηση ενώ σε περίπτωση αποτυχίας μεταβαίνει στο γέμισμα των πεδίων με τα τελευταία δεδομένα.

```
JAVASCRIPT (hub.js)

function getPositionHandler(pos) {
    indeterminate_progressbar.style.visibility = "collapse";
    var latitude = pos.coordinate.point.position.latitude;
    var longitude = pos.coordinate.point.position.longitude;
```

```
                var baseURL =
"http://api.wunderground.com/api/ee4561125223576a/conditions/hourly/forecast10day/astronomy/q/" + latitude +
"," + longitude + ".json";
                WinJS.Application.addEventListener = getWeatherJSON(baseURL);
            }
```

Σε κάθε περίπτωση ένα ολοκληρωμένο περιβάλλον διαχείρισης μηνυμάτων ενημερώνει τον χρήστη για τις διαδικασίες που εκτελούνται στο πάνω μέρος της οθόνης ή για τυχόν σφάλματα με το αντικείμενο του αναδυόμενου παραθύρου ειδοποίησης **WinJS.UI.Flyout**.

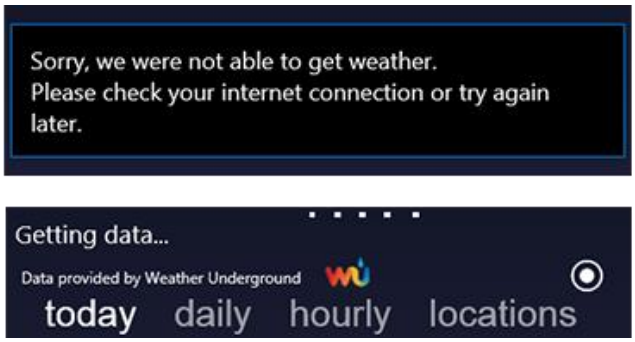


Εικόνα 3-8: Ειδοποιήσεις συνάρτησης getLoc

Ακολούθως εκτελείτε η συνάρτηση **getWeatherJSON(baseURL).** Σε όλες τις περιπτώσεις εκτέλεσής της όταν πρόκειται για νέα ενεργοποίηση, ανανέωση, κ.α. η συνάρτηση που την καλεί της δίνει και την απαραίτητη είσοδο για να λειτουργήσει. Την υλοποίηση της κλήσης αναλαμβάνει το αντικείμενο **WinJS.xhr(options).done ().** Το όρισμα options παίρνει τιμές απαραίτητες αλλά και προαιρετικές. Το απαραίτητο αντικείμενο που θα έχει πάντα είναι το url ενώ παραδείγματα προαιρετικών τιμών που δεν θα έχουμε στην δικιά μας συνάρτηση είναι η μέθοδο κλήσης **HTTP** (GET.POST ή HEAD) όπου αν παραληφθεί η προεπιλεγμένη είναι GET και τον τύπο απάντησης της κλήσης που θα δώσει ο server (response type) που το προεπιλεγμένο είναι κείμενο (String). Όταν η κλήση ολοκληρωθεί συνεχίζει με τυποποιημένες συναρτήσεις προσδιορίζοντας τι θα κάνει μετά από την επιτυχή λήψη δεδομένων ή σε περίπτωση σφάλματος και τις λειτουργίες που θα υλοποιεί κατά την διάρκεια της διαδικασίας. Κατά την διάρκεια λήψης η εφαρμογή μας ενημερώνει στο πάνω μέρος της οθόνης ότι γίνεται η λήψη δεδομένων. Αν ολοκληρωθεί σταματάει την εμφάνιση του μηνύματος, μεταφέρει το αρχείο JSON σε μία **global** μεταβλητή **data**, εκτελεί την συνάρτηση για την απόδοση των στοιχείων στις προβλεπόμενες σελίδες και στην συνέχεια με καθυστέρηση ενός δευτερολέπτου κάνει εγγραφή των νέων δεδομένων στο αρχείο μόνιμης αποθήκευσης των δεδομένων. Σε περίπτωση που αποτύχει εκτελεί την συνάρτηση **fillLastData()** χρησιμοποιώντας τα τελευταία δεδομένα για την απεικόνιση του καιρού.

```
JAVASCRIPT (hub.js)

function completed(result) {
        // handle completed download.
        if (result.status === 200) {
            indeterminate_progressbar.style.visibility = "collapse";
            var data = JSON.parse(result.responseText);
            WinJS.Application.addEventListener = fillHubSections(data, unitsVariable);
            setTimeout(function () {
                saveStateData.DATA = data;
                var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
                Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                    Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(saveStateData));
                });
            }, 1000);
        }
    },
    function error(request) {
        // handle error conditions.
        indeterminate_progressbar.style.visibility = "collapse";
        var callError = WinJS.Resources.getString('callErrorMessage');
        document.getElementById("errorMessage").winControl.show(flyoutMessage);
        document.getElementById('flyoutMessage').innerHTML = callError.value;
        setTimeout(function () {
            WinJS.Application.addEventListener = fillLastData();
        }, 1);
    });
```



Εικόνα 3-9: Ειδοποιήσεις συνάρτησης getWeatherJSON

3.3.4. Απόδοση στοιχείων από την JavaScript στην HTML

Η συνάρτηση **fillHubSections(data, unitsVariable)** υλοποιεί την απόδοση των στοιχείων προς απεικόνιση στις σελίδες. Λειτουργεί με απαραίτητη είσοδο τα δεδομένα προς απεικόνιση και την μονάδα μέτρησης, έτσι ώστε να εμφανίζει τα κατάλληλα δεδομένα. Ο σύνδεσμος για την λήψη του καιρού εμπεριέχει τα αιτήματα που έχουμε από το web service, μετά την εισαγωγή του API κλειδιού, με περιγραφική μορφή /conditions/hourly/forecast10day/astronomy/. Στην περίπτωση του παρακάτω κώδικα οι τιμές που εκχωρεί στα πεδία των σελίδων από το JSON είναι της τρέχουσας ημέρας (**conditions)** και της ώρας ανατολής/δύσης (**astronomy**). Ένα παράδειγμα προσδιορισμού σε αντικείμενο της global μεταβλητής data για την πρώτη περίπτωση

είναι η ολόκληρη περιγραφή της τοποθεσίας π.χ. **fullLocation.innerHTML = data.current_observation.display_location.full**; και για την δεύτερη περίπτωση είναι η ώρα δύσης του ηλίου π.χ. **data.moon_phase.sunset.hour**.

JAVASCRIPT (hub.js)

```
// FUNCTION - This function fill all the values of hubSection ----------------------
    function fillHubSections(data, unitsVariable) {

        // Change background day or night - Weather Icon
        var now = new Date(), sunriseHour = data.moon_phase.sunrise.hour, sunriseMinute =
data.moon_phase.sunrise.minute,
        sunsetHour = data.moon_phase.sunset.hour, sunsetMinute = data.moon_phase.sunset.minute,
currentIcon = data.current_observation.icon;

        if (((sunriseHour == now.getHours) && (sunriseMinute < now.getMinutes)) || ((sunriseHour <
now.getHours()) &&
            (now.getHours() < sunsetHour)) || ((sunsetHour == now.getHours()) && (now.getMinutes() <
sunsetMinute))) {
            document.body.style.backgroundImage = "linear-gradient(#55A9F1, #369CE2 50%, #2B82C9)";
            document.getElementById("curWeatherIcon").src = "/images/weatherIcons/" + currentIcon +
".png";
        }
        else {
            document.body.style.backgroundImage = "linear-gradient(#14172A, #39395D)";
            document.getElementById("curWeatherIcon").src = "/images/weatherIcons/nt_" + currentIcon +
".png";
        }

        // Section 1 Current Data 1 - Current Weather Data
        fullLocation.innerHTML = data.current_observation.display_location.full;
        curWeather.innerHTML = data.current_observation.weather;
        if (unitsVariable == "Celsius") {
            curTemp.innerHTML = Math.round(data.current_observation.temp_c) + "&degC";
            curFeelsLike.innerHTML = Math.round(data.current_observation.feelslike_c) + "&degC";
            curVisibility.innerHTML = data.current_observation.visibility_km + "km";
            curBarometer.innerHTML = data.current_observation.pressure_mb + "mb";
            curWindSpeed.innerHTML = data.current_observation.wind_kph + "kph";
        }
        else {
            curTemp.innerHTML = Math.round(data.current_observation.temp_f) + "&degF";
            curFeelsLike.innerHTML = Math.round(data.current_observation.feelslike_f) + "&degF";
            curVisibility.innerHTML = data.current_observation.visibility_mi + "mi";
            curBarometer.innerHTML = data.current_observation.pressure_in + "in";
            curWindSpeed.innerHTML = data.current_observation.wind_kph + "mph";
        }
        curHumidity.innerHTML = data.current_observation.relative_humidity;
        curWindDir.innerHTML = data.current_observation.wind_dir;
        curUV.innerHTML = data.current_observation.UV;
```

Η αντιστοίχιση των HTML αντικειμένων στην σχετική σελίδα έχει πραγματωθεί με την χρήση κλειδιών στις ετικέτες (π.χ. ** **) και με την εντολή **fullLocation.innerHTML** όπου καλείται να γεμίσει το πεδίο από τα δεδομένα της JavaScript.

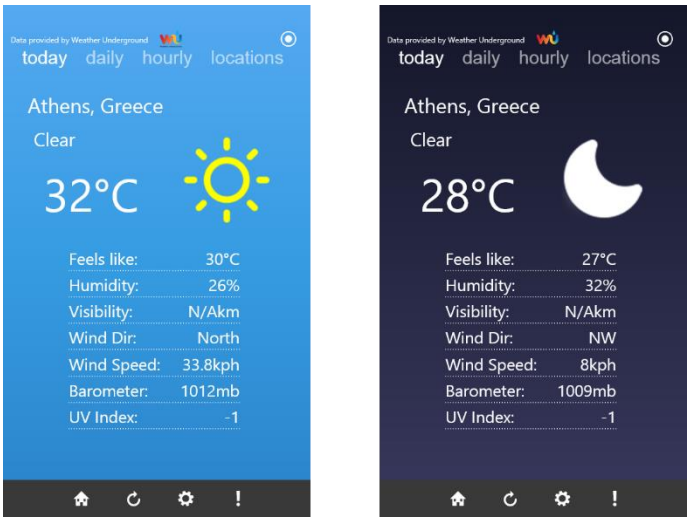
HTML (section1Page.html)

```
<div id="current_data1">
        <div id="fullLocation" class="title label-extralarge" data-win-res="{ textContent: 'location1'
}"></div>
```

```
                    <div id="current_data1b">
                      <div id="current_data1c">
                        <div id="curWeather" class="subtext" data-win-res="{ textContent: 'description1'
}"></div>
                        <div id="curTemp">?</div>
                      </div>
                      <div id="current_data1d">
                        <img src="/images/wxugLogo.png" id="curWeatherIcon" />
                      </div>
                    </div>
                  </div>

                  <div id="todayTable">
                    <div id="current_data2">
                      <table id="s1_t1">
                        <tbody>
                          <tr class="s1_t1_r">
                            <td class="s1_t1_td" data-win-res="{ textContent: 'feelslike1' }"></td>
                            <td id="curFeelsLike" class="vtd"></td>
                          </tr>
                          <tr class="s1_t1_r">
                            <td class="s1_t1_td" data-win-res="{ textContent: 'humidity1' }"></td>
                            <td id="curHumidity" class="vtd"></td>
                          </tr>
                          <tr class="s1_t1_r">
```

Επιπλέον λειτουργίες που εκτελεί είναι η αλλαγή της ταπετσαρίας αναλόγως την κατάσταση της ημέρας (μέρα ή νύχτα) και η δημιουργία των πινάκων για τα στοιχεία που πρέπει να έχει η λίστα των 7 ημερών και των 24 ωρών.



Εικόνα 3-10: Αλλαγές στην ταπετσαρία με την κατάσταση μέρα/νύχτα

Η υλοποίηση κατασκευής του πίνακα για την πρόσθεσή τους στην λίστα εβδομαδιαίας και ημερήσιας πρόβλεψης εκτελείται με μια συνάρτηση επανάληψης **for** όσο το σύνολο των ημερών και των ωρών από τα αιτήματα πρόγνωσης καιρού 10 ημερών (**forecast10day**) και 24 ωρών (**hourly**). Παράδειγμα προσδιορισμού αντικειμένου από τον παρακάτω κώδικα για την θερμοκρασία ανά ώρα σε Celsius είναι το **data.hourly_forecast[i].temp.metric;**

```
JAVASCRIPT (hub.js)

// Section 3 Hourly Data - Hourly Forecast
    var hour, hourIcon, hourTempIcon, hourTemp, newHourArray = [];
    hourListview.splice(0, 24);

    for (var i = 0; i < 24; i++) {
      hour = data.hourly_forecast[i].FCTTIME.hour_padded + ":" +
data.hourly_forecast[i].FCTTIME.min;  //15:00
      hourIcon = data.hourly_forecast[i].icon;  //clear
      if (unitsVariable == "Celsius") {
        hourTempIcon = "/images/tempIconC.png";
        hourTemp = data.hourly_forecast[i].temp.metric; //30C
      }
      else {
        hourTempIcon = "/images/tempIconF.png";  //70F
        hourTemp = data.hourly_forecast[i].temp.english;
      }

      newHourArray[i] = {
        hour: hour, hourlyicon: "/images/weatherIcons/" + hourIcon + ".png",
        hourTempIcon: hourTempIcon, hourTemp: hourTemp
      };

      hourListview.push(newHourArray[i]);

    }
```

3.3.5. Δημιουργία λίστας με το αντικείμενο WinJS.Binding.List

Η τοποθέτηση του κώδικα για τις λίστες είναι στο ίδιο χώρο με αυτόν των συναρτήσεων. Πιο αναλυτικά γίνεται η δήλωση των μεταβλητών **var dayListview;** για το αντικείμενο λίστα και **var dailyItemsArray = [ ];** για τον πίνακα με τα αντίστοιχα στοιχεία. Στην συνέχεια δημιουργείται νέα λίστα με το αντικείμενο **new WinJS.Binding.List(dailyItemsArray);** και τελειώνει με τον προσδιορισμό ενός Namespace όσον αφορά τα στοιχεία που θα έχει η λίστα (π.χ. **data: dayListview**).

```
JAVASCRIPT (hub.js)

// CREATE DAILY LISTVIEW ----------------------------------------------
  var dayListview;
  var dailyItemsArray = [];
  dayListview = new WinJS.Binding.List(dailyItemsArray);
  WinJS.Namespace.define("Daily.ListView", {
    data: dayListview
  });
  WinJS.UI.processAll();
```

Ο απαιτούμενος κώδικας για την απόδοση των στοιχείων στις ετικέτες HTML είναι ο ίδιος με τον παραπάνω. Αυτό που αλλάζει στις λίστες είναι ο προσδιορισμός ιδιοτήτων που υλοποιούνται με το **data-win-control** για να αποκτήσει το αντικείμενο τις ιδιότητες χειρισμού μιας λίστας **WinJS.UI.ListView** και το **data-win-options** για να αποκτήσει τις επιλογές μιας λίστας όπως το **itemDataSource** για τον προσδιορισμό της πηγής των αντικειμένων της λίστας, το **itemTemplate** για τον προσδιορισμό της ετικέτας html που θα χρησιμοποιήσει για την απεικόνιση (π.χ. **<div**

class="dailyItemTemplate">), το **selectionMode** για την τρόπο επιλογής των αντικειμένων της λίστας, το **tapBehavior** για την συμπεριφορά της λίστας στο πάτημα ενός αντικειμένου της και τέλος το **layout** για τον προσδιορισμό τύπου της λίστας ως προς τον τρόπο απεικόνισης της.

HTML (section2Page.html)

```
<!-- Daily ListView-->
        <div class="dailyItemTemplate" data-win-control="WinJS.Binding.Template" style="display:
none">
          <div class="dailyTemplatecontainer">
            <div id="dailyCell" data-win-bind="textContent: day"></div>
            <div id="weekDay" data-win-bind="textContent: weekday"></div>
            <div id="dailyIconCell"><img src="#" class="dailyIcon" data-win-bind="src: dailyIcon"
/></div>
            <div id="dailyTempIconCell"><img src="#" class="dailyListTempIcon" data-win-
bind="src: dailyTempIcon" /></div>
            <div id="dailyMinMaxTemp">
              <span data-win-bind="textContent: maxTemp"></span><br />
              <span data-win-bind="textContent: minTemp"></span>
            </div>
            <div id="dailyHumiIconCell"><img src="/images/humiIcon.png"
class="dailyListHumiIcon" /></div>
            <div id="dailyHumi" data-win-bind="textContent: humi"></div>
          </div>
        </div>

        <!-- The declarative markup necesary for ListView instantiation -->
        <!-- Call WinJS.UI.processAll() in your initialization code -->
        <div class="listView win-selectionstylefilled"
          data-win-control="WinJS.UI.ListView"
          data-win-options="{
          itemDataSource: Daily.ListView.data.dataSource,
          itemTemplate: select('.dailyItemTemplate'),
          selectionMode: WinJS.UI.SelectionMode.none,
          tapBehavior: WinJS.UI.TapBehavior.none,
          layout: { type: WinJS.UI.ListLayout }
          }">
        </div>
```

Ένα παράδειγμα κώδικα στυλ μορφοποίησης μιας λίστας αναγράφεται παρακάτω. Ο τρόπος εμφάνισης δεδομένων στην λίστα είναι με πλέγμα (grid) και ο προσδιορισμός των αντίστοιχων αντικειμένων HTML γίνεται με την ιδιότητα id. Συγκεκριμένα αυτό γίνεται θέτοντας το αντικείμενο της HTML ** ** σε απεικόνιση **display: -ms-grid;** στην συνέχεια το χωρίζεις σε γραμμές **-ms-grid-rows** και στήλες **-ms-grid-columns** και τέλος προσδιορίζεις τα εμφωλευμένα αντικείμενα HTML σε ποια γραμμή και στήλη θα απεικονίζονται.

CSS (hubPhone.css)

```
.hubpage .hub .section2 .dailyTemplatecontainer {
        display: -ms-grid;
        -ms-grid-columns: 12% 15% 31% 11% 11% 7% 10%;
        font-weight: lighter;
```

```
                font-size: 24px;
                line-height: 26px;
                text-align: center;
                width: 90%;
                background-image: url('/images/listBG.png');
}

#dailyCell {
                -ms-grid-column: 1;
                height: 26px;
                margin: auto 0;
}

#weekDay {
                -ms-grid-column: 2;
                height: 26px;
                margin: auto 0;
                text-transform: uppercase;
}

#dailyIconCell {
                -ms-grid-column: 3;
                height:90px;
                margin: auto 0;
                text-align: right;
}
```

3.3.6. Διαχείριση εντολών των κουμπιών

Κάθε φορά που η σελίδα **hub.html** ολοκληρώνει την ετοιμασία των δεδομένων της προς κοινοποίηση, αλλάζει σε κατάσταση ready. Σε αυτό το σημείο γίνεται ο προσδιορισμός των συναρτήσεων που θα εκτελούν τα διάφορα αντικείμενα HTML (π.χ. button) των σελίδων. Δίνοντας στην ετικέτα **button** την ιδιότητα κλειδιού **id** η γλώσσα JavaScript συνδυάζει αυτό το κουμπί με τις αντίστοιχες εντολές που θα πρέπει να εκτελέσει και υλοποιείται με τον κώδικα **document.queryselector(“#id”).addEventListener(“click”, function, false);**

Παρακάτω αναγράφεται ο κώδικας HTML για τον προσδιορισμό του αντικειμένου μπάρας στο κάτω μέρος της οθόνης (**WinJS.UI.AppBar**) με τον προσδιορισμό επιλογών ενός τέτοιου αντικειμένου με την επιλογή **id** για την ιδιότητα ταυτότητας του κουμπιού, **label** για την ετικέτα εμφάνισης και icon για το εικονίδιο που θα έχει.

HTML (default.html)

```
<div id="appBar" class="minimalAppBar" data-win-control="WinJS.UI.AppBar">
      <button data-win-control="WinJS.UI.AppBarCommand"
data-win-options="{id:'cmdHome',label:'Home',icon:'home',tooltip:'Home'}"></button>
```

Στην συνέχεια αναγράφεται ο κώδικας διαχείρισης συναρτήσεων των κουμπιών και των αντίστοιχων συναρτήσεων που εκτελούν. Στο σημείο των συναρτήσεων να σημειώσουμε το αντικείμενο της WinJS για την περιήγηση στις σελίδες της εφαρμογής **WinJS.Navigation.navigate("/settings/settings.html");**

```
JAVASCRIPT (hub.js)

// Event handlers for appbar
        document.querySelector("#cmdHome").addEventListener("click", homeWeather, false);
        document.querySelector("#cmdRefresh").addEventListener("click", refreshWeather, false);
        document.querySelector("#cmdSettings").addEventListener("click", navSettings, false);
        document.querySelector("#cmdAbout").addEventListener("click", navAbout, false);
```

```
JAVASCRIPT (hub.js)

// FUNCTIONS - APPBAR ---------------------------------------------------------------------
   function homeWeather() {
      var baseURL = HomeLocationURL;
      WinJS.Application.addEventListener = getWeatherJSON(baseURL);
   }
   function refreshWeather() {
      var baseURL = lastBaseURL;
      WinJS.Application.addEventListener = getWeatherJSON(baseURL);
   }
   function navSettings() {
      WinJS.Navigation.navigate("/settings/settings.html");
   }
   function navAbout() {
      WinJS.Navigation.navigate("/settings/about.html");
   }
```

3.3.7. Γραμμή αναζήτησης με AutoSuggestBox Control

Η γραμμή αναζήτησης με την χρήση του αντικειμένου AutoSuggestBox έχει την ιδιότητα αναζήτησης αποτελεσμάτων αυτόματα με το γράψιμο κάθε νέου χαρακτήρα. Ο κώδικας JavaScript βρίσκεται στο αρχείο **data.js** μαζί με την λίστα τοποθεσιών μετατρέποντας την συνεργασία αυτών των δύο αντικειμένων σε μεγαλύτερης απόδοσης. Ο κώδικας HTML για την προσθήκη ενός τέτοιου αντικειμένου υλοποιείται με την εισαγωγή ιδιότητας **data-win-control="WinJS.UI.AutoSuggestBox”** στο αντικείμενο της HMTL και τις επιλογές **placeholderText** για το κείμενο που θα βρίσκεται στην γραμμή αναζήτησης, **onsuggestionsrequested** για την αναζήτηση και ετοιμασία της λίστας αποτελεσμάτων και **onquerysubmitted** για διαχείριση συμβάντων κατά την πληκτρολόγηση.

```
HTML (section4Page.html)

<div id="searchBox" class="box" style="display: none">
        <div data-win-control="WinJS.UI.AutoSuggestBox" data-win-options="{
            placeholderText: 'Type a city',
            onsuggestionsrequested: LocationAutoSearchBox.suggestionsRequestedHandler,
            onquerysubmitted: LocationAutoSearchBox.querySubmittedHandler
          }"></div>
        <div id="status"></div>
    </div>
```

Η αντίστοιχη JavaScript αυτού του αντικειμένου δηλώνοντας τις απαραίτητες μεταβλητές για την εκτέλεση των συναρτήσεων και προσδιορίζοντας το αντικείμενο

με το **WinJS.Namespace.define** συνεχίζει την λειτουργία του με την πρώτη συνάρτηση να παίρνει το κείμενο από την γραμμή αναζήτησης, να κάνει κλήση στο web service με το αντίστοιχο κατασκευασμένο σύνδεσμο και να εμφανίζει τα αποτελέσματα και την άλλη να διαχειρίζεται τα συμβάντα.

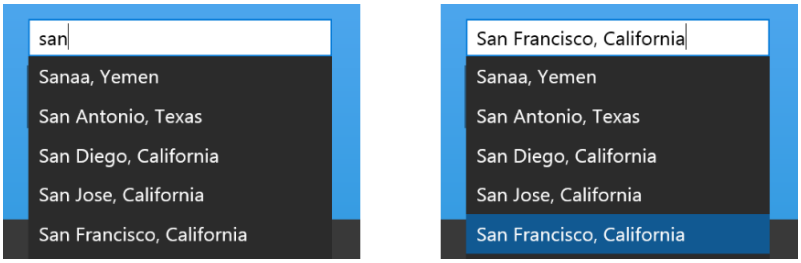
JAVASCRIPT (data.js)

```
//FUNCTIONS - AUTOSUGGESTBOX ------------------------------------
  var statusEl = document.getElementById("status");
  var xhrRequest;

  function suggestionsRequestedHandler(eventObject) {
    var queryText = eventObject.detail.queryText,
      query = queryText.toLowerCase(),
    suggestionCollection = eventObject.detail.searchSuggestionCollection;
    if (queryText.length > 0) {
      var suggestionUri = "http://autocomplete.wunderground.com/aq?query=" + query;
      // Cancel the previous suggestion request if it is not finished.
      if (xhrRequest && xhrRequest.cancel) {
        xhrRequest.cancel();
      }
      // Create request to obtain suggestions from the web service.
      xhrRequest = WinJS.xhr({ url: suggestionUri });
      eventObject.detail.setPromise(xhrRequest.then(
        function (request) {
          if (request.responseText) {
            var parsedResponse = JSON.parse(request.responseText);
            for (var i = 0, len = parsedResponse.RESULTS.length; i < len; i++) {
              suggestionCollection.appendQuerySuggestion(parsedResponse.RESULTS[i].name);
            }
          }
        }));
    }
  }
  function querySubmittedHandler(eventObject) {
    var queryText = eventObject.detail.queryText;
    WinJS.log && WinJS.log(queryText, "sample", "status");
  }

  WinJS.log = function (msg, source, type) {
    if (type === "status") {
      WinJS.Application.addEventListener = addLocation(msg);
      searchBox.style.display = "none";
    }
  }
  WinJS.Namespace.define("LocationAutoSearchBox", {
    suggestionsRequestedHandler: WinJS.UI.eventHandler(suggestionsRequestedHandler),
    querySubmittedHandler: WinJS.UI.eventHandler(querySubmittedHandler)
  });

  WinJS.UI.processAll();
```



Εικόνα 3-11: Παραδείγματα χρήσης γραμμής αναζήτησης

3.3.8. Διαδραστική λίστα τοποθεσιών

Τα στοιχεία που αφορούν τις αλλαγές στην μόνιμη αποθήκευση έχουν τοποθετηθεί ανεξάρτητα έτσι ώστε να μην συγχέονται οι αποθηκεύσεις στα δεδομένα μιας και είναι όλα στο ίδιο αρχείο. Στις περιπτώσεις που το JSON έχεις απευθείας δύο στοιχεία (π.χ. **"SETTINGS": { "unitsVariable": "Celsius", "launchLocationVariable": "HomeLocation" }**) δημιουργείται ένα αντικείμενο με αυτά τα δύο στοιχεία και με την προσθήκη ενημερώνεται.

Στην περίπτωση των τοποθεσιών το αρχείο είναι σε μορφή πίνακα με αποτέλεσμα η εισαγωγή και διαγραφή να πρέπει να γίνει με κλειδί. Έτσι έχοντας δηλώσει την σελίδα **data.js** στην head της default.html και της hub.html η επικοινωνία τους καθίσταται εφικτή. Η data επειδή εκτελείται πρώτη σε σειρά στην εκκίνηση διαβάζει το αρχείο με τις τοποθεσίες, δημιουργεί τον πίνακα και δίνει τα αντικείμενα στην λίστα. Οι επιπλέον επιλογές που έχει η συγκεκριμένη λίστα στο control της είναι το **selectionMode** με τιμή **single** και χρησιμοποιείται στο σενάριο επιλογής ενός αντικειμένου από πολλά άλλα, το **tapBehavior** με τιμή **directSelect** δίνοντας συμπεριφορά στο πάτημα ενός αντικειμένου και του **footer** επιλέγοντας το με το όνομα της κλάσης **select('.footer')**. Επίσης έχουν προστεθεί και επιλογές της λίστας στην σελίδα **hub.html** για να μπορεί να διαχειριστεί την ομαλή περιήγηση μέσα στον κόμβο με τις 4 σελίδες και τα δεδομένα αυτής. Οι επιλογές είναι το **datasource** όπου μέσα από ένα άλλο αντικείμενο της **hub.js** ορίζει ποια θα είναι τα δεδομένα της λίστας και το **oniteminvoked** που ορίζει ποια θα είναι η ενέργεια στο άγγιγμα αντικειμένου.

HTML (hub.html)

```
dataSource: select('.pagecontrol').winControl.LocationsDataSource,
layout: {type: WinJS.UI.ListLayout},
oniteminvoked: select('.pagecontrol').winControl.LocationsItemMenu
```

Ο προσδιορισμός των παραπάνω επιλογών για τα δεδομένα και το μενού που θα εμφανίζεται ελέγχεται από κώδικα που βρίσκεται μετά την κατάσταση ready της σελίδας hub.html και τα αντικείμενα που ζητάει από την data είναι δηλωμένα εκεί.

JAVASCRIPT (hub.js)

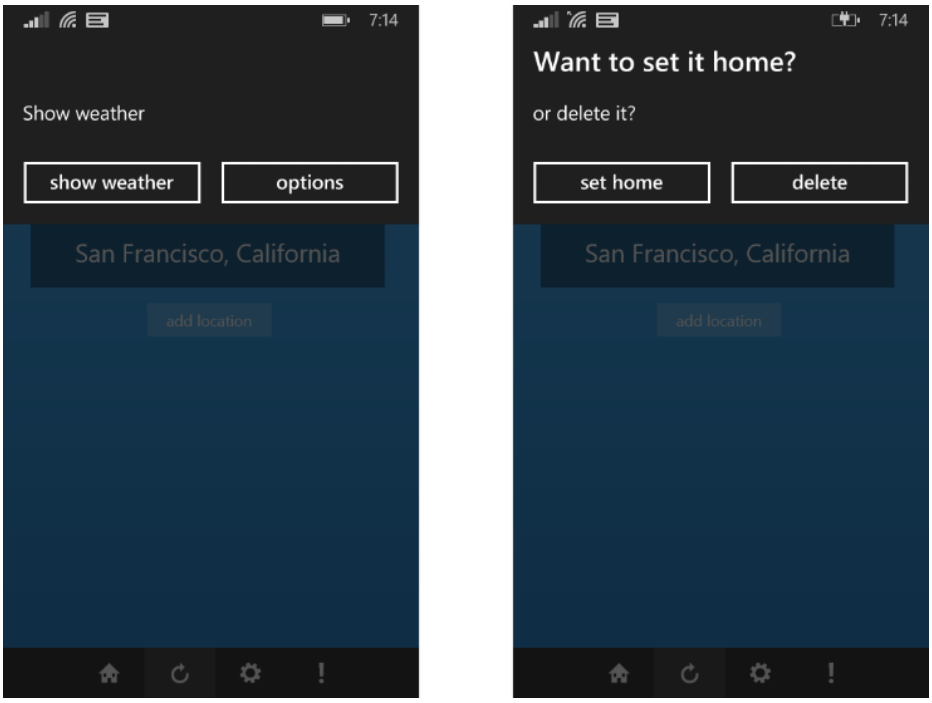
```
LocationsDataSource: LocationsItems.dataSource,
LocationsItemMenu: util.markSupportedForProcessing(function (args) {
    var menu = Data.openMenu(args.detail.itemIndex, navigate);
    if (navigate == true) {
        var callnNavigate = Data.takeURL(URL);
        WinJS.Application.addEventListener = nav.navigate("/pages/hub/hub.html");
        setTimeout(function () {
            WinJS.Application.addEventListener = getWeatherJSON(baseURL);
        }, 100);
```

```
        }
    })
```

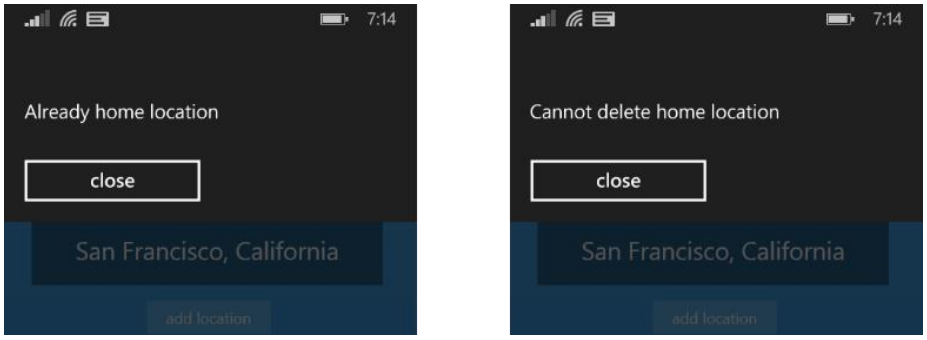
Το αντικείμενο WinJS.Namespace.define δημιουργεί έναν έλεγχο για όλη την σελίδα και τις συναρτήσεις της. Τα αριστερά μέρη της δηλώσεων αφορούν τα αιτήματα της σελίδας hub.js και τα δεξιά την διαχείριση των αντίστοιχων συναρτήσεων στην σελίδα. Η πρώτη είναι αυτή που επιστρέφει τα δεδομένα που διαβάστηκαν από την μόνιμη αποθήκευση, η δεύτερη μεταφέρει τα αντικείμενα της λίστα προς εμφάνιση, η τρίτη είναι αυτή που αποδίδει το κλειδί ενός αντικειμένου της λίστας αφού επιλεχθεί η τέταρτη είναι η συνάρτηση που ανοίγει το παράθυρο διαλόγων/μενού για την επιλογή αντίστοιχης εντολής.

JAVASCRIPT (data.js)

```
WinJS.Namespace.define("Data", {
    savedData: transferDataToHub,
    getItemsFromLocations: getItemsForLocations,
    getItemReference: getItemReference,
    openMenu: openMenu,
    takeURL: takeURL
});
```



Εικόνα 3-12: Κεντρικό μήνυμα μετά από επιλογή τοποθεσίας – επιλογές



Εικόνα 3-13: Μηνύματα από αποσφαλμάτωσης στις επιλογές τοποθεσίας

3.3.9. Πολλυγλωσικός μηχανισμός

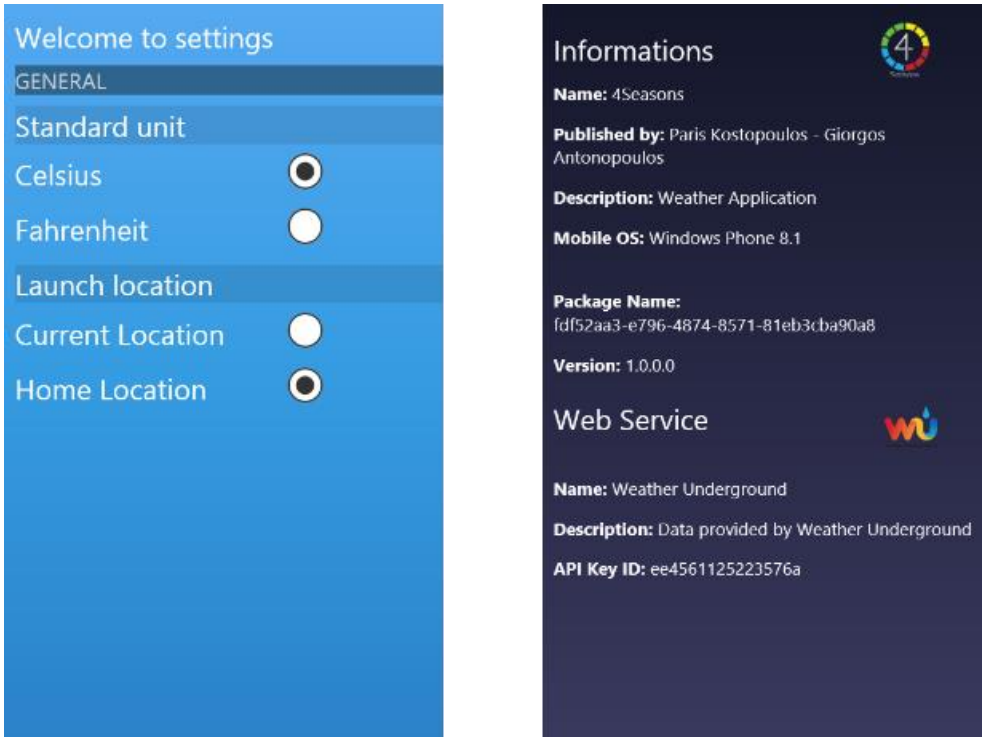
Οι Windows Store εφαρμογές διαθέτουν ένα σύστημα επιλογής γλωσσών από το **package.appxmanifest** σύμφωνα με την γλώσσα του κινητού και με την επιλογή της μέσα από την ίδια την εφαρμογή. Στο μονοπάτι των αρχείων **strings/en-US/** υπάρχει το αρχείο **resources.resjson** όπου κρατάει όλες τις λέξεις που εμφανίζονται στην εφαρμογή και καλούνται μέσω του κώδικα **var message = WinJS.Resources.getString('someMessage')** και εμφανίζονται σε ένα HTML αντικείμενο με τον κώδικα **progress.innerHTML = message.value.** Ένα διαφορετικό τρόπο χρήσης των λέξεων χρησιμοποιεί η φόρτωση των τίτλων hunSections και αυτός είναι με το **data-win-control** του αντικειμένου **WinJS.UI.PivotItem** και με την επιλογή **data-win-res="{ winControl: {'header': 'Section1'}.**

RESJSON (resources.resjson)

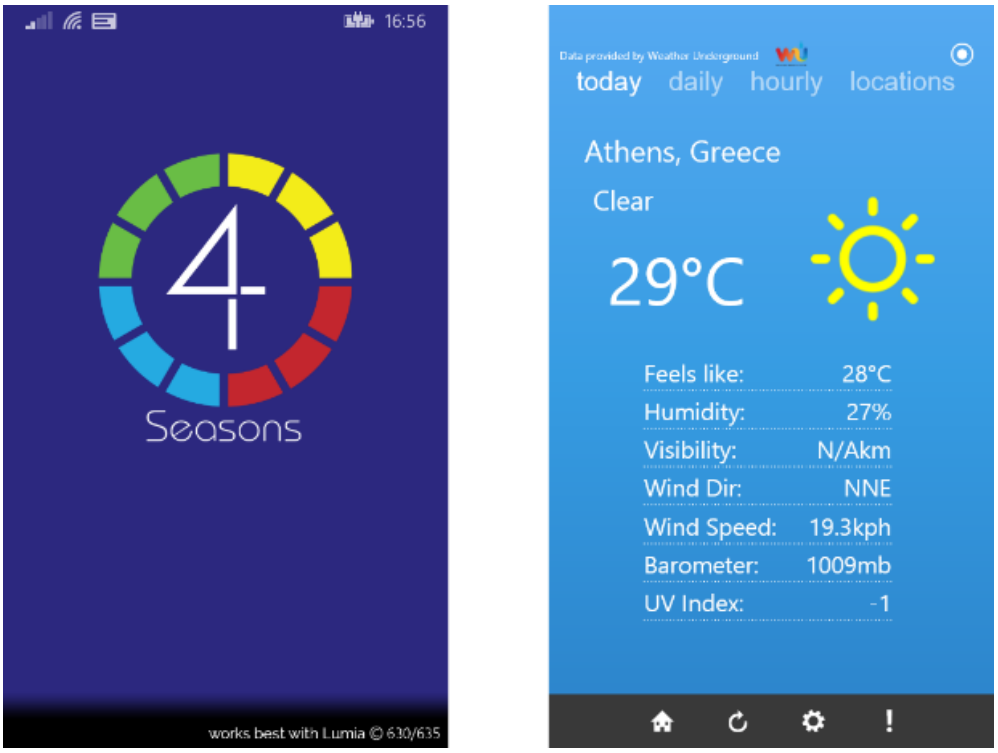
```
{
        // Section 1 - Today
        "Section1": "today",
        "fullLocation": "Location",
        "description": "Description",
        "feelslike1": "Feels like:",
        "humidity1": "Humidity:",
        "uvindex1": "UV Index:",
        "visibility1": "Visibility:",
        "barometer1": "Barometer:",
        "winddirection1": "Wind Dir:",
        "windspeed1": "Wind Speed:",
        // Section 2 - Daily
        "Section2": "daily",
          "dayTapDescription": "Select a day for more information",
        // Section 3 - Hourly
          "Section3": "hourly",
        // Section 4 - Locations

        // Settings Page
        "sHeader": "Welcome to settings",
        "sGeneral": "General",
        "sUnits": "Standard unit",
        "CLabel": "Celsi
}
```

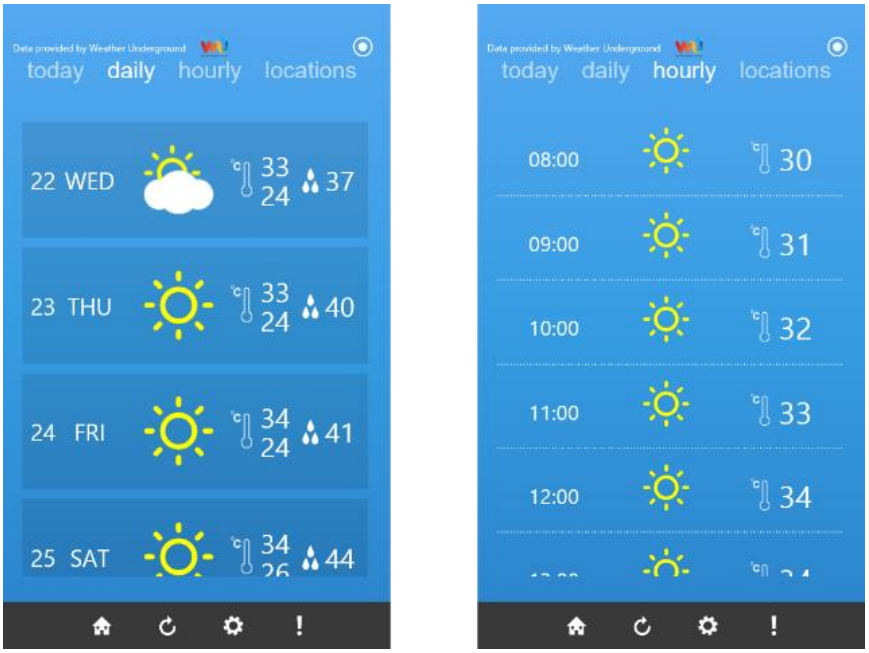
3.3.10. Άλλα παραδείγματα χρήσης



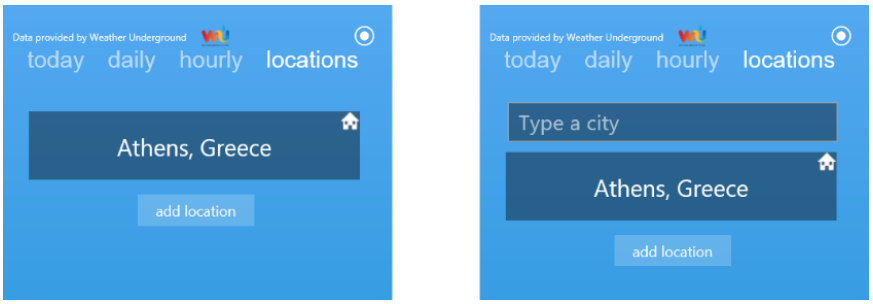
Εικόνα 3-14: Σελίδες settings και about



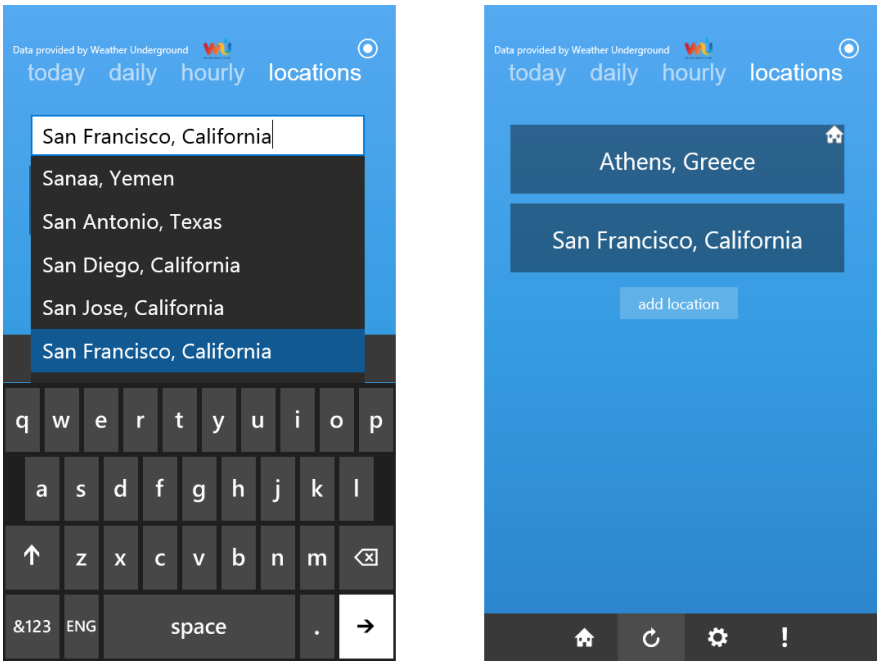
Εικόνα 3-15: Παράδειγμα σελίδων 1 της εφαρμογής την μέρα



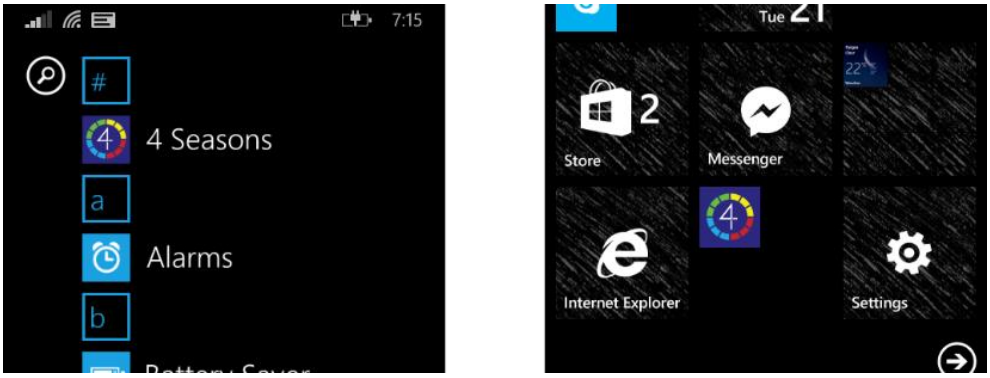
Εικόνα 3-16: Παράδειγμα σελίδων 2 της εφαρμογής την μέρα



Εικόνα 3-17: Παράδειγμα χρήσης της σελίδας τοποθεσιών



Εικόνα 3-18: Παράδειγμα προσθήκης τοποθεσίας



Εικόνα 3-19: Πλακίδια εφαρμογής 1

Εικόνα 3-20: Πλακίδια εφαρμογής 2

ΚΕΦΑΛΑΙΟ 4. ΣΥΜΠΕΡΑΣΜΑΤΑ

4.1. Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως στόχο την ανάπτυξη μιας εφαρμογής καιρού για το λειτουργικό σύστημα Windows Phone 8.1 καθώς και την παροχή γνώσεων σχετικά με την δημιουργία της. Έτσι μέσα από τα κεφάλαια της εργασίας έγινε γνωστό τι είναι smartphone, λειτουργικά συστήματα των έξυπνων κινητών τηλεφώνων, το ποσοστό τους στην παγκόσμια αγορά καθώς και αναφορά στις απαραίτητες τεχνολογίες για την λειτουργία της εφαρμογής. Στην συνέχεια έγινε μια εισαγωγή στο λειτουργικό σύστημα Windows Phone, στις εκδόσεις του, τα χαρακτηριστικά του και την αρχιτεκτονική του. Επιπλέον, έγιναν καταγραφή των τεχνολογιών που χρησιμοποιήθηκαν για τη δημιουργία και την ανάπτυξη της εφαρμογής.

Με την καταγραφή σχετικών τεχνολογιών και την υλοποίηση της εφαρμογής, η κατανόηση του προβλήματος που τέθηκε ως πτυχιακή εργασία έγινε πιο ακριβής. Ακόμα έγιναν σαφή τα χαρακτηριστικά που θα πρέπει να έχει μία εφαρμογή όπως ορθότητα λειτουργίας εφόσον συμπεριφέρεται σύμφωνα με τις λειτουργικές τις απαιτήσεις, αξιοπιστία με το να μην υπάρχει ανώμαλη λειτουργία σε περίπτωση λάθους, αποδοτικότητα που έχει να κάνει με την καλή σύνταξη κώδικα και με την συμπεριφορά του λειτουργικού που έχεις επιλέξει, τις γλώσσες ανάπτυξης και τους πόρους των συσκευών που προορίζεται και ευχρηστία έχοντας ένα φιλικό περιβάλλον στον χρήστη.

Πιο συγκεκριμένα, η εφαρμογή «4Seasons» αποτελεί μια αξιόπιστη πηγή για την απεικόνιση του καιρού από ένα σύγχρονο web service και το επιθυμητό αποτέλεσμα που δείχνει την ικανοποίηση του χρήστη είναι η δημιουργία ανάγκης για μελλοντική επέκταση.

4.2. Μελλοντικές επεκτάσεις

Ο ρυθμός εξέλιξης της τεχνολογίας είναι γρήγορος, με αποτέλεσμα να βγαίνουν στο φως καθημερινά νέες τεχνολογίες και δυνατότητες. Είναι επόμενο συνεπώς να παρουσιαστούν μελλοντικές επεκτάσεις της εφαρμογής «4Seasons». Δύναται να συμπεριλάβει απεικόνιση του καιρού σε χάρτες για περισσότερες πληροφορίες, σχετικά με την οποιαδήποτε τοποθεσία επιθυμεί ο χρήστης. Επιπλέον, δύναται η επέκταση για αποστολή/κοινοποίηση πληροφοριών, φωτογραφιών ακόμα και βίντεο

από τα καιρικά φαινόμενα που επικρατούν σε κάποια τοποθεσία ή για κάποιο πρόβλημα που δημιουργήθηκε λόγω ακραίων καιρικών φαινομένων (π.χ. σε οδικές αρτηρίες). Επίσης θα μπορούσε να προστεθεί στην εφαρμογή ένα μικρό παράθυρο αξιολόγησης κατά την έξοδο του χρήστη με ένα μηχανισμό αξιολόγησης και ένα προαιρετικό σχόλιο.

Επιπρόσθετα θα μπορούσε να γίνει πρόσθεση νέων γλωσσών, η ζωντανή απεικόνιση καιρού στα πλακίδια της εφαρμογής (Live Tiles), η επέκταση της εφαρμογής σε Universal App έχοντας την δυνατότητα εγκατάστασης/χρήσης και σε άλλες συσκευές (laptop, tablet) και η αποθήκευση τελευταίων δεδομένων για κάθε αποθηκευμένη περιοχή και όχι μόνο της τελευταίας κλήσης, δημιουργώντας για κάθε μια από αυτές ένα αρχείο JSON με την χώρα και την πόλη (AthensGR.json). Αυτό δίνει στον χρήστη το πλεονέκτημα να δει τα τελευταία δεδομένα σε ειδικές περιπτώσεις (π.χ. ο χρήστης δεν έχει internet) ή σε περίπτωση ανάγκης (π.χ. πρόβλημα στο δίκτυο λόγω κακοκαιρίας).

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ελληνόγλωσσες αναφορές

(1) James F.Kurose και Keith W.Ross, 2008, *Δικτύωση Υπολογιστών Προσέγγιση από πάνω προς τα κάτω (Τέταρτη έκδοση)*, Εκδόσεις: Μ.Γκιούρδας, Αθήνα.

Ελληνόγλωσσες αναφορές

(2) Brockschmidt Kraig, 2014, *Programming Windows Store Apps with HTML, CSS, and JavaScript (Second Edition)*, Published by Microsoft Press, Redmond, σελ. 128-131

Διαδικτυακές αναφορές

(Οι διαθέσιμες ιστοσελίδες τοποθετούνται βάσει κεφαλαίων)

(3) http://www.phonescoop.com/glossary/term.php?gid=131
(4) https://el.wikipedia.org/wiki/Smartphone
(5) http://kynourianet.gr/σαν-σήμερα-06031983-κυκλοφορεί-το-πρώτο-κινη/
(6) http://www.sansimera.gr/articles/241
(7) http://www.idc.com/prodserv/smartphone-market-share.jsp
(8) https://el.wikipedia.org/wiki/Λειτουργικό_σύστημα
(9) http://www.gameworld.gr/component/content/article/188-ειδήσεις/2014-05-19-05-49-58/17483-2011-04-04-10-52-43
(10) http://www.idc.com/prodserv/smartphone-os-market-share.jsp
(11) https://en.wikipedia.org/wiki/App_store
(12) http://www.smartinsights.com/mobile-marketing/app-marketing/mobile-app-statistics/
(13) https://en.wikipedia.org/wiki/Mobile_Web
(14) https://el.wikipedia.org/wiki/Κυψελωτό_δίκτυο
(15) https://en.wikipedia.org/wiki/2G
(16) https://el.wikipedia.org/wiki/Global_System_for_Mobile_Communications
(17) https://www.daniweb.com/software-development/computer-science/threads/35959/evolution-from-3g-to-4g-and-beyond-5g
(18) https://en.wikipedia.org/wiki/3G#cite_note-ITU-1
(19) https://en.wikipedia.org/wiki/4G
(20) https://el.wikipedia.org/wiki/IEEE_802.11
(21) https://en.wikipedia.org/wiki/IEEE_802.11ac
(22) http://arstechnica.com/business/2009/12/wifi-looks-to-1-gigabit-horizon/1/
(23) http://eu.mio.com/el_gr/global-positioning-system_how-does-gps-work.htm
(24) http://www.walkhighlands.co.uk/safety/gps-smartphones.shtml
(25) https://el.wikipedia.org/wiki/Global_Positioning_System
(26) http://www.it.uom.gr/project/soap/Theory/introduction.html
(27) http://www.it.uom.gr/project/soap/Theory/architecture.html
(28) https://el.wikipedia.org/wiki/Διεπαφή_προγραμματισμού_εφαρμογών

(29) http://domainresellers.gr/what-is-api.php
(30) http://www.bloggінghits.com/2013/07/04/the-brief-history-of-windows-phone/
(31) http://www.pestaola.gr/windows-phone-8-1-features/
(32) http://www.windowsphone.com/en-in/how-to/wp8/basics/whats-new-in-windows-phone
(33) https://el.wikipedia.org/wiki/Αρχιτεκτονική_ARM
(34) http://www.jayway.com/2014/04/14/windows-phone-8-1-for-developers-for-the-htmljavascript-developer/
(35) https://software.intel.com/en-us/blogs/2013/08/19/use-html5-device-orientation-to-run-windows-8-javascript-apps-on-android-devices
(36) https://en.wikipedia.org/wiki/JavaScript
(37) https://en.wikipedia.org/wiki/HTML
(38) https://en.wikipedia.org/wiki/Cascading_Style_Sheets
(39) https://en.wikipedia.org/wiki/JSON
(40) https://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Express
(41) https://en.wikipedia.org/wiki/Microsoft_Blend
(42) https://msdn.microsoft.com/en-us/library/windows/apps/dn412707.aspx
(43) https://msdn.microsoft.com/en-us/library/windows/apps/dn615002.aspx
(44) https://msdn.microsoft.com/en-us/library/windows/apps/hh770579.aspx
(45) https://msdn.microsoft.com/en-us/library/windows/apps/jj126146.aspx
(46) http://www.w3schools.com/js/js_scope.asp

Έγγραφο από τοποθεσία Web

(47) Bret Stateham, aka.ms, Building Windows 8 Apps using HTML5/CSS3/JS, http://aka.ms/w8js , Win8 HTML5 CSS3 JS/Slides/ Building Windows 8 Apps using HTML5-JS-CSS.pptx, 2013, Microsoft Developer Evangelist

Βιβλιογραφία εικόνων

(Η βιβλιογραφία των εικόνων είναι διαθέσιμη στις παρακάτω ιστοσελίδες)

(48) https://dsepwiki.wikispaces.com/2η+γενιά+κινητών+τηλεφώνων
(49) http://www.gsmarena.com/apple_iphone_3g-pictures-2424.php
(50) http://www.randomheads.com/techhead/comparing-existing-smartphones-market/2015/04/03/
(51) http://www.intechopen.com/books/trends-in-telecommunications-technologies/performance-analysis-of-time-of-arrival-mobile-positioning-in-wireless-cellular-cdma-networks
(52) http://tech360.co.il/gsm/
(53) http://www.theregister.co.uk/2011/05/24/wtf_is_4g/
(54) http://www.tvtechnology.com/video-networking/0152/uncompressed-wireless-hd/213236
(55) http://www.info-gadget.science/2014/07/perbedaan-wifi-80211abgnac.html
(56) http://www.service.architecture.com/articles/webservices/application_program_interfaces_apis.html
(57) http://www.wunderground.com/

ΠΑΡΑΡΤΗΜΑ Ι – ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

default.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>_4Seasons</title>

    <!-- WinJS references -->
    <!-- At runtime, ui-themed.css resolves to ui-themed.light.css or ui-themed.dark.css
    based on the user's theme setting. This is part of the MRT resource loading functionality. -->
    <link href="winjs/css/ui-dark.css" rel="stylesheet" />
    <script src="winjs/js/base.js"></script>
    <script src="winjs/js/ui.js"></script>

    <!-- PhonePivotApp references -->
    <link href="/WinJS/css/phone.css" rel="stylesheet" />
    <script src="/js/data.js"></script>
    <script src="/js/navigator.js"></script>
    <script src="/js/default.js"></script>
</head>
<body class="phone">

    <div id="contenthost" data-win-control="Application.PageControlNavigator" data-win-
options="{home: '/pages/hub/hub.html'}"></div>

    <div id="appBar" class="minimalAppBar" data-win-control="WinJS.UI.AppBar">
        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'cmdHome',label:'Home',icon:'home',tooltip:'Home'}"></button>
        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'cmdRefresh',label:'Refresh',icon:'refresh',tooltip:'Refresh'}"></button>
        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'cmdSettings',label:'Settings',icon:'settings',tooltip:'Settings'}"></button>
        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'cmdAbout',label:'About',icon:'important',tooltip:'About'}"></button>
    </div>

</body>
</html>
```

default.js

```
// For an introduction to the Pivot template, see the following documentation:
// http://go.microsoft.com/fwlink/?LinkID=392284
(function () {
    "use strict";

    var activation = Windows.ApplicationModel.Activation;
    var app = WinJS.Application;
    var nav = WinJS.Navigation;
    var sched = WinJS.Utilities.Scheduler;
    var ui = WinJS.UI;

    app.addEventListener("activated", function (args) {
        if (args.detail.kind === activation.ActivationKind.launch) {
            if (args.detail.previousExecutionState !== activation.ApplicationExecutionState.terminated) {
                // TODO: This application has been newly launched. Initialize
                // your application here.
            } else {

            };

            hookUpBackButtonGlobalEventHandlers();
```

```
                nav.history = app.sessionState.history || {};
                nav.history.current.initialPlaceholder = true;

                // Optimize the load of the application and while the splash screen is shown, execute high priority
scheduled work.
                ui.disableAnimations();
                var p = ui.processAll().then(function () {
                    return nav.navigate(nav.location || Application.navigator.home, nav.state);
                }).then(function () {
                    return sched.requestDrain(sched.Priority.aboveNormal + 1);
                }).then(function () {
                    ui.enableAnimations();
                });

                args.setPromise(p);
            }
        });

        app.oncheckpoint = function (args) {
            // TODO: This application is about to be suspended. Save any state
            // that needs to persist across suspensions here. If you need to
            // complete an asynchronous operation before your application is
            // suspended, call args.setPromise().
            app.sessionState.history = nav.history;
        };

        function hookUpBackButtonGlobalEventHandlers() {
            // Subscribes to global events on the window object
            window.addEventListener('keyup', backButtonGlobalKeyUpHandler, false)
        }

        // CONSTANTS
        var KEY_LEFT = "Left";
        var KEY_BROWSER_BACK = "BrowserBack";
        var MOUSE_BACK_BUTTON = 3;

        function backButtonGlobalKeyUpHandler(event) {
            // Navigates back when (alt + left) or BrowserBack keys are released.
            if ((event.key === KEY_LEFT && event.altKey && !event.shiftKey && !event.ctrlKey) ||
(event.key === KEY_BROWSER_BACK)) {
                nav.back();
            }
        }

        app.start();
    })();
```

data.js

```
    (function () {
        "use strict";

        var locationsList = new WinJS.Binding.List();
        var locationItems = [];
        var saveStateData;

        // Our Location Data List
        var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
        Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
            Windows.Storage.FileIO.readTextAsync(file).then(function (text) {
                saveStateData = JSON.parse(text);
                for (var i = 0, len = saveStateData.LOCATIONS.length; i < len; i++) {
                    var name = saveStateData.LOCATIONS[i].name;
                    var img = saveStateData.LOCATIONS[i].img;
                    locationItems[i] = { name: name, img: img };
                }
```

```
                locationItems.forEach(function (item) {
                    locationsList.push(item);
                });
            });
        });

        WinJS.Namespace.define("Data", {
            savedData: transferDataToHub,
            getItemsFromLocations: getItemsForLocations,
            getItemReference: getItemReference,
            openMenu: openMenu,
            takeURL: takeURL
        });

        function takeURL () {
            return baseURL;
        }

        var baseURL;
        function openMenu(itemIndex, navigate) {
            var navigate = false;
            var saveStateData;
            var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
            Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                Windows.Storage.FileIO.readTextAsync(file).then(function (text) {
                    saveStateData = JSON.parse(text);
                    var askname = saveStateData.LOCATIONS[itemIndex].name;
                });
            });

            var msg = new Windows.UI.Popups.MessageDialog("Show weather");

            // SHOW WEATHER
            msg.commands.append(new Windows.UI.Popups.UICommand("show weather", function
(command) {
                baseURL = "http://api.wunderground.com/api/ee4561125223576a/" +
                "conditions/hourly/forecast10day/astronomy/q/" + saveStateData.LOCATIONS[itemIndex].name
+ ".json";
                navigate = true;
                return navigate;
            }));

            // SHOW OPTION
            msg.commands.append(new Windows.UI.Popups.UICommand("options", function (command) {
                var options = new Windows.UI.Popups.MessageDialog("or delete it?", "Want to set it home?");

                // SET HOME OPTION
                options.commands.append(new Windows.UI.Popups.UICommand("set home", function
(command) {
                    if (saveStateData.LOCATIONS[itemIndex].home == false) {
                        for (var i = 0, len = saveStateData.LOCATIONS.length; i < len; i++) {
                            saveStateData.LOCATIONS[i].home = false;
                            saveStateData.LOCATIONS[i].img = "/images/emptyflag.png";
                        }
                        saveStateData.LOCATIONS[itemIndex].home = true;
                        saveStateData.LOCATIONS[itemIndex].img = "/images/homeflag.png";
                        Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                            Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(saveStateData));
                        });
                        for (var i = 0, len = saveStateData.LOCATIONS.length; i < len; i++) {
                            var name = saveStateData.LOCATIONS[i].name;
                            var img = saveStateData.LOCATIONS[i].img;
                            locationItems[i] = { name: name, img: img };
                        }
                        locationsList.splice(0, len);
                        locationItems.forEach(function (item) {
                            locationsList.push(item);
```

```
                        });
                    } else {
                        var newMsg = new Windows.UI.Popups.MessageDialog("Already home location");
                        newMsg.showAsync();
                    }
                }));

                // DELETE OPTION
                options.commands.append(new Windows.UI.Popups.UICommand("delete", function (command)
{
                    if (saveStateData.LOCATIONS[itemIndex].home == false) {
                        locationsList.splice(itemIndex, 1);
                        delete saveStateData.LOCATIONS[itemIndex];
                        Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                            Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(saveStateData));
                        });
                    } else {
                        var newMsg = new Windows.UI.Popups.MessageDialog("Cannot delete home location");
                        newMsg.showAsync();
                    }
                }));
                options.showAsync();
                return navigate;
            }));

            // Set the command that will be invoked by default
            msg.defaultCommandIndex = 1;

            // Show the message dialog
            msg.showAsync();
        }

        //Transfer already read saved data to hub.js
        function transferDataToHub() {
            return saveStateData;
        }
        //Returns Location list
        function getItemsForLocations() {
            return locationsList;
        }
        // Get a reference for an item, using as key the item title
        function getItemReference(item) {
            return [item.name];
        }

        //Add new Location
        function addLocation(msg) {
            var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
            var refreshedData;
            Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                Windows.Storage.FileIO.readTextAsync(file).then(function (text) {
                    refreshedData = JSON.parse(text);
                    for (var i = 0, len = refreshedData.LOCATIONS.length; i < len; i++) {
                        if (msg != refreshedData.LOCATIONS[i].name) {
                            var addnew = true;
                        }
                        else {
                            addnew = false;
                            break;
                        }
                    }
                    if (addnew == true) {
                        var object = { name: msg, home: false, img: "/images/emptyflag.png" };
                        locationsList.push(object);
                        Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                            Windows.Storage.FileIO.readTextAsync(file).then(function (text) {
                                refreshedData = JSON.parse(text);
```

```
                                refreshedData['LOCATIONS'].push({ "name": msg, "home": false });
                                Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function
(file) {
                                    Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(refreshedData));
                                });
                            });
                        });
                    }
                    else {
                        var cityexist = new Windows.UI.Popups.MessageDialog("This city already exists");
                        cityexist.showAsync();
                    }
                });
            });
        }

        //FUNCTIONS - AUTOSUGGESTBOX ---------------------------------------------------------------------------
-----------------------
        var statusEl = document.getElementById("status");
        var xhrRequest;

        function suggestionsRequestedHandler(eventObject) {
            var queryText = eventObject.detail.queryText,
                query = queryText.toLowerCase(),
            suggestionCollection = eventObject.detail.searchSuggestionCollection;
            if (queryText.length > 0) {
                var suggestionUri = "http://autocomplete.wunderground.com/aq?query=" + query;
                // Cancel the previous suggestion request if it is not finished.
                if (xhrRequest && xhrRequest.cancel) {
                    xhrRequest.cancel();
                }
                // Create request to obtain suggestions from the web service.
                xhrRequest = WinJS.xhr({ url: suggestionUri });
                eventObject.detail.setPromise(xhrRequest.then(
                    function (request) {
                        if (request.responseText) {
                            var parsedResponse = JSON.parse(request.responseText);
                            for (var i = 0, len = parsedResponse.RESULTS.length; i < len; i++) {
                                suggestionCollection.appendQuerySuggestion(parsedResponse.RESULTS[i].name);
                            }
                        }
                    }));
            }
        }
        function querySubmittedHandler(eventObject) {
            var queryText = eventObject.detail.queryText;
            WinJS.log && WinJS.log(queryText, "sample", "status");
        }
        WinJS.log = function (msg, source, type) {
            if (type === "status") {
                WinJS.Application.addEventListener = addLocation(msg);
                searchBox.style.display = "none";
            }
        }
        WinJS.Namespace.define("LocationAutoSearchBox", {
            suggestionsRequestedHandler: WinJS.UI.eventHandler(suggestionsRequestedHandler),
            querySubmittedHandler: WinJS.UI.eventHandler(querySubmittedHandler)
        });

        WinJS.UI.processAll();
    })();
```

phone.css

```
    .phone {
            background-image: linear-gradient(#14172A, #272844 50%, #39395D);
```

```
}

#contenthost {
	background-color: gray;
}

#indeterminate_progressbar {
	visibility: collapse;
}

#progressbar {
	width: 100%;
	color: #fff;
	height: 4px;
	margin-top: 2px;
	display: block;
}

#progress {
	line-height: 22px;
	font-weight: lighter;
	height: 22px;
	position: absolute;
	top: 6px;
	left: 6px;
}

.fragment header[role=banner] {
	/* Define a grid with columns for the back button and page title. */
	-ms-grid-columns: 1fr;
	-ms-grid-rows: 1fr;
	display: -ms-grid;
	height: 40px;
}

	.fragment header[role=banner] .win-navigation-backbutton {
		visibility: hidden;
	}

	.fragment header[role=banner] .backbutton {
		visibility: hidden;
	}

	.fragment header[role=banner] .titlearea {
		-ms-grid-column: 1;
		margin-top: 10px;
		margin-left: 20px;
		font-size: 20pt;
		overflow: visible;
	}

		fragment header[role=banner] .win-navigation-backbutton {
		visibility: hidden;
	}

	.fragment header[role=banner] .backbutton {
		visibility: hidden;
	}

	.fragment header[role=banner] .titlearea {
		-ms-grid-column: 1;
		margin-top: 10px;
		margin-left: 10px;
		font-size: 20pt;
		overflow: visible;
	}
```

```
.fragment header[role=banner] .titlearea .pagetitle {
        width: calc(100% - 20px);
}

#topView {
        margin-top: 10px;
}

#topView a {
        font-size: 10px;
        font-weight: lighter;
        color: white;
        text-decoration: none;
        top: -17px;
        position: relative;
        left: 10px;
}

#wuLogo {
        width: 30px;
        position: relative;
        left: 20px;
        top: -8px;
}

#getLocation {
        left: 120px;
}

#cmdAbout {
        margin-right: 15%;
}

.win-toggleswitch-value {
        font-size: 24px !important;
}

.searchbtn {
        width: 90%;
        text-align: center;
        margin-top: 10px;
}
```

hub.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>hubPage</title>

   <link href="/WinJS/css/phone.css" rel="stylesheet" />
   <link href="/pages/hub/hubphone.css" rel="stylesheet" />
   <script src="/js/data.js"></script>
   <script src="/pages/hub/hub.js"></script>
   <script src="/pages/hub/section1Page.js"></script>
   <script src="/pages/hub/section2Page.js"></script>
   <script src="/pages/hub/section3Page.js"></script>
   <script src="/pages/hub/section4Page.js"></script>
</head>
<body>

   <div class="hubpage fragment">

      <div id="indeterminate_progressbar">
         <progress id="progressbar" class="win-progress-bar"></progress>
```

```
                    <div id="progress"></div>
                </div>

                <header aria-label="Header content" role="banner">
                    <button class="titlearea backbutton" data-win-control="WinJS.UI.BackButton"></button>
                    <div id="topView">
                        <a href="http://www.wunderground.com">Data provided by Weather Underground</a>
                        <img src="/images/wxugLogo.png" id="wuLogo" alt="Weather Underground Logo" />
                        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'getLocation',icon:'target'}"></button>
                    </div>
                </header>

                <section aria-label="Main content" role="main">
                    <!-- Customize the Hub control by modifying the HubSection controls here. -->

                    <div class="hub" data-win-control="WinJS.UI.Pivot">

                        <div class="section1 section" data-win-control="WinJS.UI.PivotItem" data-win-options="{
isHeaderStatic: true }" data-win-res="{ winControl: {'header': 'Section1'} }">
                            <div class="sectioncontrol" id="section1contenthost" data-win-
control="HubApps_SectionControls.Section1Control"></div>
                        </div>

                        <div class="section2 section" data-win-control="WinJS.UI.PivotItem" data-win-options="{
isHeaderStatic: true }" data-win-res="{ winControl: {'header': 'Section2'} }">
                            <div class="sectioncontrol" id="section2contenthost" data-win-
control="HubApps_SectionControls.Section2Control"></div>
                        </div>

                        <div class="section3 section" data-win-control="WinJS.UI.PivotItem" data-win-options="{
isHeaderStatic: true }" data-win-res="{ winControl: {'header': 'Section3'} }">
                            <div class="section3contenthost sectioncontrol" id="section3contenthost" data-win-
control="HubApps_SectionControls.Section3Control"></div>
                        </div>

                        <div class="section4 section" data-win-control="WinJS.UI.PivotItem" data-win-options="{
isHeaderStatic: true }" data-win-res="{ winControl: {'header': 'Section4'} }">
                            <div class="section4contenthost sectioncontrol" id="section4contenthost" data-win-
control="HubApps_SectionControls.Section4Control"
                                data-win-options="{
                                    dataSource: select('.pagecontrol').winControl.LocationsDataSource,
                                    layout: {type: WinJS.UI.ListLayout},
                                    oniteminvoked: select('.pagecontrol').winControl.LocationsItemMenu
                                }"></div>
                        </div>
                    </div>

                    <div id="errorMessage" data-win-control="WinJS.UI.Flyout"><div
id="flyoutMessage"></div></div>

                </section>
            </div>
        </body>
        </html>
```

hub.js

```
        (function () {
            "use strict";

            var nav = WinJS.Navigation;
            var session = WinJS.Application.sessionState;
            var util = WinJS.Utilities;

            var saveStateData = Data.savedData();
```

```
            var LocationsItems = Data.getItemsFromLocations();

            // GLOBAL VARIABLES ----------------------------------------------------------------------------------------
---------------
            // If app is activated
            var activated = false;
            // Home Location baseURL
            var HomeLocationURL;
            var FavoriteLocationURL;
            // Refresh URL
            var lastBaseURL = HomeLocationURL;
            // Units ( Celsius - Fahrenheit )
            var unitsVariable;
            // Launch Location ( CurrentLocation - HomeLocation )
            var launchLocationVariable;
            // Data for getWeatherJSON
            var data;

            // HUB PAGE Define -----------------------------------------------------------------------------------------
----------
            WinJS.UI.Pages.define("/pages/hub/hub.html", {
                processed: function (element) {
                    //TODO: Start the weather app (Only for the app activation)
                    if (activated == false) {
                        WinJS.Application.addEventListener = readSettingsData();
                    }
                    return WinJS.Resources.processAll(element);
                },

                // This function is called whenever a user navigates to this page. It
                // populates the page elements with the app's data.
                ready: function (element, options) {
                    var hub = element.querySelector(".hub").winControl;
                    hub.onheaderinvoked = function (args) {
                        args.detail.section.onheaderinvoked(args);
                    };
                    hub.onloadingstatechanged = function (args) {
                        if (args.srcElement === hub.element && args.detail.loadingState === "complete") {
                            hub.onloadingstatechanged = null;
                            hub.element.focus();
                        }
                    }

                    //When navigate back to hub
                    if (activated == true) {
                        setTimeout(function () {
                            WinJS.Application.addEventListener = fillHubSections(data, unitsVariable);
                        }, 1);
                    }

                    // Button Handlers
                    // Event handler for the get location icon
                    document.querySelector("#getLocation").addEventListener("click", getLoc, false);

                    // Event handlers for appbar
                    document.querySelector("#cmdHome").addEventListener("click", homeWeather, false);
                    document.querySelector("#cmdRefresh").addEventListener("click", refreshWeather, false);
                    document.querySelector("#cmdSettings").addEventListener("click", navSettings, false);
                    document.querySelector("#cmdAbout").addEventListener("click", navAbout, false);

                },

                LocationsDataSource: LocationsItems.dataSource,

                LocationsItemMenu: util.markSupportedForProcessing(function (args) {
                    var menu = Data.openMenu(args.detail.itemIndex, navigate);
                    if (navigate == true) {
```

```
                    var callnNavigate = Data.takeURL(URL);
                    WinJS.Application.addEventListener = nav.navigate("/pages/hub/hub.html");
                    setTimeout(function () {
                        WinJS.Application.addEventListener = getWeatherJSON(baseURL);
                    }, 100);
                }
            }),

            unload: function () {
                // TODO: Respond to navigations away from this page.
            },

            updateLayout: function (element) {
                /// <param name="element" domElement="true" />

                // TODO: Respond to changes in layout.
            },
        });

        // SETIINGS PAGE Define ----------------------------------------------------------------------------------------
------------
        WinJS.UI.Pages.define("/settings/settings.html", {
            // This function is called whenever a user navigates to this page. It
            // populates the page elements with the app's data.
            ready: function (element, options) {
                appBar.style.visibility = "collapse";
                // TODO: Initialize the page here.
                document.getElementById(unitsVariable).checked = true;
                document.getElementById(launchLocationVariable).checked = true;
                WinJS.Resources.processAll();
            },

            unload: function () {
                appBar.style.visibility = "visible";
                //TODO: Respond to navigations away from this page.
                WinJS.Application.addEventListener = SaveSettings();
            }
        });

        // FUNCTIONS ---------------------------------------------------------------------------------------------------
---------
        // LOAD LAST SAVED SETTINGS
        var saveStateData;
        var HomeLocationURL;
        function readSettingsData() {
            unitsVariable = saveStateData.SETTINGS.unitsVariable;
            launchLocationVariable = saveStateData.SETTINGS.launchLocationVariable;
            data = saveStateData.DATA;
            for (var i = 0, len = saveStateData.LOCATIONS.length; i < len; i++) {
                if (saveStateData.LOCATIONS[i].home == true) {
                    HomeLocationURL =
"http://api.wunderground.com/api/ee4561125223576a/conditions/hourly/forecast10day/astronomy/q/" +
saveStateData.LOCATIONS[i].name + ".json";
                }
            }
            activated = true;
            WinJS.Application.addEventListener = startWeather();
        }

        // FUNCTION - This function call weather JSON by the ckecked launchLocationVariable on settings -
----------------------------
        function startWeather() {
            if (launchLocationVariable == "CurrentLocation") {
                WinJS.Application.addEventListener = getLoc();
            } else {
                var baseURL = HomeLocationURL;
                setTimeout(function () {
```

```
                        WinJS.Application.addEventListener = getWeatherJSON(baseURL);
                    }, 1);
                }
            }

            // FUNCTION - This function detect the user's location ----------------------------------------------------------
---------
            var loc = null;
            function getLoc() {
                // MESSAGE - START
                indeterminate_progressbar.style.visibility = "visible";
                var progressMessage = WinJS.Resources.getString('gettingLocation');
                progress.innerHTML = progressMessage.value;
                if (loc == null) {
                    loc = new Windows.Devices.Geolocation.Geolocator();
                }
                if (loc != null) {
                    loc.getGeopositionAsync().then(getPositionHandler, errorHandler);
                }
            }

            function getPositionHandler(pos) {
                indeterminate_progressbar.style.visibility = "collapse";
                var latitude = pos.coordinate.point.position.latitude;
                var longitude = pos.coordinate.point.position.longitude;
                var baseURL =
"http://api.wunderground.com/api/ee4561125223576a/conditions/hourly/forecast10day/astronomy/q/" + latitude +
"," + longitude + ".json";
                WinJS.Application.addEventListener = getWeatherJSON(baseURL);
            }

            function errorHandler(e) {
                indeterminate_progressbar.style.visibility = "collapse";
                document.getElementById("errorMessage").winControl.show(flyoutMessage);
                document.getElementById('flyoutMessage').innerHTML = getStatusString(loc.locationStatus);
                WinJS.Application.addEventListener = fillLastData();
            }

            function getStatusString(locStatus) {
                switch (locStatus) {
                    case Windows.Devices.Geolocation.PositionStatus.ready:
                        // Location data is available
                        return "Location is available.";
                        break;
                    case Windows.Devices.Geolocation.PositionStatus.initializing:
                        // This status indicates that a GPS is still acquiring a fix
                        return "A GPS device is still initializing.";
                        break;
                    case Windows.Devices.Geolocation.PositionStatus.noData:
                        // No location data is currently available
                        return "Data from location services is currently unavailable.";
                        break;
                    case Windows.Devices.Geolocation.PositionStatus.disabled:
                        // The app doesn't have permission to access location,
                        // either because location has been turned off.
                        return "Your location is currently turned off. " +
                            "Change your settings through the Settings charm " +
                            " to turn it back on.";
                        break;
                    case Windows.Devices.Geolocation.PositionStatus.notInitialized:
                        // This status indicates that the app has not yet requested
                        // location data by calling GetGeolocationAsync() or
                        // registering an event handler for the positionChanged event.
                        return "Location status is not initialized because " +
                            "the app has not requested location data.";
                        break;
                    case Windows.Devices.Geolocation.PositionStatus.notAvailable:
```

```
                        // Location is not available on this version of Windows
                        return "You do not have the required location services " +
                            "present on your system.";
                        break;
                    default:
                        break;
                }
            }

            // FUNCTION - This function call weather JSON - conditions/hourly/forecast10day/astronomy - by
baseURL ----------------------
            function getWeatherJSON(baseURL) {
                indeterminate_progressbar.style.visibility = "visible";
                var progressMessage = WinJS.Resources.getString('gettingData');
                progress.innerHTML = progressMessage.value;
                lastBaseURL = baseURL;
                WinJS.xhr({
                    url: baseURL
                }).done(
                function completed(result) {
                    // handle completed download.
                    if (result.status === 200) {
                        indeterminate_progressbar.style.visibility = "collapse";
                        var data = JSON.parse(result.responseText);
                        WinJS.Application.addEventListener = fillHubSections(data, unitsVariable);
                        setTimeout(function () {
                            saveStateData.DATA = data;
                            var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
                            Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                                Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(saveStateData));
                            });
                        }, 1000);
                    }
                },
                function error(request) {
                    // handle error conditions.
                    indeterminate_progressbar.style.visibility = "collapse";
                    var callError = WinJS.Resources.getString('callErrorMessage');
                    document.getElementById("errorMessage").winControl.show(flyoutMessage);
                    document.getElementById('flyoutMessage').innerHTML = callError.value;
                    setTimeout(function () {
                        WinJS.Application.addEventListener = fillLastData();
                    }, 1);
                });
            }

            // FUNCTION - This function fill all the values of hubSection ----------------------------------------------
---------------
            function fillHubSections(data, unitsVariable) {

                // Change background day or night - Weather Icon
                var now = new Date(), sunriseHour = data.moon_phase.sunrise.hour, sunriseMinute =
data.moon_phase.sunrise.minute,
                sunsetHour = data.moon_phase.sunset.hour, sunsetMinute = data.moon_phase.sunset.minute,
currentIcon = data.current_observation.icon;

                if (((sunriseHour == now.getHours) && (sunriseMinute < now.getMinutes)) || ((sunriseHour <
now.getHours()) &&
                    (now.getHours() < sunsetHour)) || ((sunsetHour == now.getHours()) && (now.getMinutes() <
sunsetMinute))) {
                    document.body.style.backgroundImage = "linear-gradient(#55A9F1, #369CE2 50%, #2B82C9)";
                    document.getElementById("curWeatherIcon").src = "/images/weatherIcons/" + currentIcon +
".png";
                }
                else {
                    document.body.style.backgroundImage = "linear-gradient(#14172A, #39395D)";
```

```
                document.getElementById("curWeatherIcon").src = "/images/weatherIcons/nt_" + currentIcon +
".png";
            }

            // Section 1 Current Data 1 - Current Weather Data
            fullLocation.innerHTML = data.current_observation.display_location.full;
            curWeather.innerHTML = data.current_observation.weather;
            if (unitsVariable == "Celsius") {
                curTemp.innerHTML = Math.round(data.current_observation.temp_c) + "&degC";
                curFeelsLike.innerHTML = Math.round(data.current_observation.feelslike_c) + "&degC";
                curVisibility.innerHTML = data.current_observation.visibility_km + "km";
                curBarometer.innerHTML = data.current_observation.pressure_mb + "mb";
                curWindSpeed.innerHTML = data.current_observation.wind_kph + "kph";
            }
            else {
                curTemp.innerHTML = Math.round(data.current_observation.temp_f) + "&degF";
                curFeelsLike.innerHTML = Math.round(data.current_observation.feelslike_f) + "&degF";
                curVisibility.innerHTML = data.current_observation.visibility_mi + "mi";
                curBarometer.innerHTML = data.current_observation.pressure_in + "in";
                curWindSpeed.innerHTML = data.current_observation.wind_kph + "mph";
            }
            curHumidity.innerHTML = data.current_observation.relative_humidity;
            curWindDir.innerHTML = data.current_observation.wind_dir;
            curUV.innerHTML = data.current_observation.UV;

            // Section 2 Daily Data - Daily Forecast
            var day, weekDay, dayIcon, dailyTempIcon, dayMaxTemp, dayMinTemp, dayHumidity,
newDayArray = [];
            dayListview.splice(0, 8);

            for (var i = 1; i < 8; i++) {
                day = data.forecast.simpleforecast.forecastday[i].date.day;  //18
                weekDay = data.forecast.simpleforecast.forecastday[i].date.weekday_short;  //Thu
                dayIcon = data.forecast.simpleforecast.forecastday[i].icon;  //tstorms
                dayHumidity = data.forecast.simpleforecast.forecastday[i].avehumidity;  //71

                if (unitsVariable == "Celsius") {
                    dailyTempIcon = "/images/tempIconC.png";
                    dayMaxTemp = data.forecast.simpleforecast.forecastday[i].high.celsius;  //27
                    dayMinTemp = data.forecast.simpleforecast.forecastday[i].low.celsius;  //18
                }
                else {
                    dailyTempIcon = "/images/tempIconF.png";
                    dayMaxTemp = data.forecast.simpleforecast.forecastday[i].high.fahrenheit;  //80
                    dayMinTemp = data.forecast.simpleforecast.forecastday[i].low.fahrenheit;  //65
                }

                newDayArray[i] = {
                    day: day, weekday: weekDay, dailyIcon: "/images/weatherIcons/" + dayIcon + ".png",
                    dailyTempIcon: dailyTempIcon, maxTemp: dayMaxTemp, minTemp: dayMinTemp, humi:
dayHumidity
                };

                dayListview.push(newDayArray[i]);

            }

            // Section 3 Hourly Data - Hourly Forecast
            var hour, hourIcon, hourTempIcon, hourTemp, newHourArray = [];
            hourListview.splice(0, 24);

            for (var i = 0; i < 24; i++) {
                hour = data.hourly_forecast[i].FCTTIME.hour_padded + ":" +
data.hourly_forecast[i].FCTTIME.min;  //15:00
                hourIcon = data.hourly_forecast[i].icon;  //clear
                if (unitsVariable == "Celsius") {
                    hourTempIcon = "/images/tempIconC.png";
```

```
                    hourTemp = data.hourly_forecast[i].temp.metric; //30C
                }
                else {
                    hourTempIcon = "/images/tempIconF.png";  //70F
                    hourTemp = data.hourly_forecast[i].temp.english;
                }

                newHourArray[i] = {
                    hour: hour, hourlyicon: "/images/weatherIcons/" + hourIcon + ".png",
                    hourTempIcon: hourTempIcon, hourTemp: hourTemp
                };

                hourListview.push(newHourArray[i]);

            }

        }

        //FUNCTIONS - CREATE DAILY LISTVIEW - DAILY HANDLER ------------------------------------
-----------------------------------
        var dayListview;
        var dailyItemsArray = [];
        dayListview = new WinJS.Binding.List(dailyItemsArray);
        WinJS.Namespace.define("Daily.ListView", {
            data: dayListview
        });
        WinJS.UI.processAll();

        //FUNCTIONS - CREATE HOYRLY LISTVIEW ----------------------------------------------------------
---------------------------
        var hourListview;
        var hourlyItemsArray = [];
        hourListview = new WinJS.Binding.List(hourlyItemsArray);
        WinJS.Namespace.define("Hourly.ListView", {
            data: hourListview
        });
        WinJS.UI.processAll();

        // FUNCTIONS - APPBAR ----------------------------------------------------------------------------
--------------
        function homeWeather() {
            var baseURL = HomeLocationURL;
            WinJS.Application.addEventListener = getWeatherJSON(baseURL);
        }
        function refreshWeather() {
            var baseURL = lastBaseURL;
            WinJS.Application.addEventListener = getWeatherJSON(baseURL);
        }
        function navSettings() {
            WinJS.Navigation.navigate("/settings/settings.html");
        }
        function navAbout() {
            WinJS.Navigation.navigate("/settings/about.html");
        }

        // SETTINGS FUNCTIONS ----------------------------------------------------------------------------
---------------
        function SaveSettings() {
            unitsVariable = document.querySelector("input[name=unitsVar]:checked").value;
            launchLocationVariable = document.querySelector("input[name=LaunchLoc]:checked").value;
            var object = { unitsVariable: unitsVariable, launchLocationVariable: launchLocationVariable };
            saveStateData.SETTINGS = object;
            var url = new Windows.Foundation.Uri("ms-appx:///saveState.json");
            Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function (file) {
                Windows.Storage.FileIO.writeTextAsync(file, JSON.stringify(saveStateData));
            });
        }
```

```
        //FUNCTIONS - Fill Last Data ----------------------------------------------------------------------------------
        function fillLastData() {
            setTimeout(function () {
                WinJS.Application.addEventListener = fillHubSections(data, unitsVariable);
            }, 200);
        }

        // ABOUT PAGE Define ----------------------------------------------------------------------------------------
------------
        WinJS.UI.Pages.define("/settings/about.html", {
            ready: function (element, options) {
                appBar.style.visibility = "collapse";
            },
            unload: function () {
                appBar.style.visibility = "visible";
            }
        })

    })();
```

hubPhone.css

```
.win-pivot .win-pivot-header {
    height: 32px;
}

.win-pivot button.win-pivot-header,
.win-pivot button.win-pivot-header:hover,
.win-pivot button.win-pivot-header:hover:active {
        margin: 0px;
}

.win-pivot .win-pivot-surface {
        height: 78vh;
}

.win-pivot .win-pivot-viewport {
        padding-top: 0px;
        margin-top: 0px;
}

.win-pivot-item .win-pivot-item-content {
        padding: 0px;
}

.hubpage header[role=banner] {
    position: relative;
    z-index: 2;
}

.hubpage section[role=main] {
    -ms-grid-row: 1;
    -ms-grid-row-span: 2;
    z-index: 1;
}

.hubpage .hub {
        margin-top: 4px;
        height: 100%;
}

        .hubpage .hub .win-hub-surface {
                height: 100%;
        }
```

```
        /* ------------ SECTION 1 ------------------------------------------------------------------------------------------
*/

                .hubpage .hub .section1 {
                        width: 95vw;
                        top: 8px;
                }

                        #current_data1 {
                                display: -ms-grid;
                                -ms-grid-rows: 40px 1fr;
                                -ms-grid-columns: 1fr;
                        }

                        .hubpage .hub .section1 .title {
                                font-size: 20pt;
                                font-weight: lighter;
                                -ms-grid-column: 1;
                                -ms-grid-row: 1;
                                margin-left: 12px;
                        }

                        #current_data1b {
                                -ms-grid-column: 1;
                                -ms-grid-row: 2;
                                display: -ms-grid;
                                -ms-grid-columns: 50% 1fr;
                                -ms-grid-rows: 1fr;
                                margin-top: 8px;
                        }

                        #current_data1c {
                                -ms-grid-column: 1;
                                -ms-grid-row: 1;
                                display: -ms-grid;
                                -ms-grid-columns: 1fr;
                                -ms-grid-rows: 26px 1fr;
                        }

                        .hubpage .hub .section1 .subtext {
                                font-size: 18pt;
                                font-weight: lighter;
                                -ms-grid-column: 1;
                                -ms-grid-row: 1;
                                margin-left: 20px;
                        }

                        #curTemp {
                                -ms-grid-column: 1;
                                -ms-grid-row: 2;
                                font-size: 60px;
                                font-weight: lighter;
                                line-height: 62px;
                                height: 44pt;
                                margin: auto 0;
                                text-align: center;
                        }

                        #current_data1d {
                                -ms-grid-column: 2;
                                -ms-grid-row: 1;
                                margin-left: 12px;
                        }

                        #curWeatherIcon {
                                width: 140px;
                                height: 140px;
```

```
                        }

                        #todayTable {
                                margin-top:10px;
                                margin-left: 0px;
                                font-size: 16pt;
                                font-weight: lighter;
                        }

                        #current_data2 {
                                width: 60%;
                                margin-left: 16.5%;
                        }

                        #current_data2 td {
                                border-bottom: 2px dotted white;
                        }

                        #s1_t1 {
                                width: 100%;
                        }

                        .s1_t1_r {
                                height: 32px;
                        }

                        .s1_t1_td {
                                width: 60%;
                        }

                        .vtd {
                                text-align: right;
                                width: 100%;
                        }

        /* ------------- SECTION 2 ---------------------------------------------------------------------------------
*/

                .hubpage .hub .section2 {
                        top: 10px;
                        height: 100%;
                }

                .dayFullInfo {
                        margin-bottom: 8px;
                        margin-top: 2px;
                }

                        .hubpage .hub .section2 .win-container {
                                margin-bottom: 0px;
                                margin-left: 0px;
                                margin-right: 0px;
                        }

                        .hubpage .hub .section2 .dailyTemplatecontainer {
                                display: -ms-grid;
                                -ms-grid-columns: 12% 15% 31% 11% 11% 7% 10%;
                                font-weight: lighter;
                                font-size: 24px;
                                line-height: 26px;
                                text-align: center;
                                width: 90%;
                                background-image: url('/images/listBG.png');
                        }

                        #dailyCell {
                                -ms-grid-column: 1;
```

```
        height: 26px;
        margin: auto 0;
}

#weekDay {
        -ms-grid-column: 2;
        height: 26px;
        margin: auto 0;
        text-transform: uppercase;
}

#dailyIconCell {
        -ms-grid-column: 3;
        height:90px;
        margin: auto 0;
        text-align: right;
}

#dailyIconCell img {
        height: 90px;
        width: auto;
}

#dailyTempIconCell {
        -ms-grid-column: 4;
        height: 120px;
        text-align: center;

}

#dailyTempIconCell img {
        width: auto;
        margin-top: 36px;
        height: 42px;
}

#dailyMinMaxTemp {
        -ms-grid-column: 5;
        font-weight: lighter;
        font-size: 28px;
        line-height: 30px;
        height: 60px;
        margin: auto 0;
        text-align: left;
}

#dailyHumiIconCell {
        -ms-grid-column: 6;
        height: 26px;
        margin: auto 0                  }

#dailyHumiIconCell img {
        width: 16px;
        height: 26px;
}

#dailyHumi {
        -ms-grid-column: 7;
        font-size: 28px;
        line-height: 30px;
        height: 30px;
        margin: auto 0;
        text-align: center;
}
```

```
        /* ------------- SECTION 3 ------------------------------------------------------------------------------------------
*/

                .hubpage .hub .section3 {
                        top: 10px;
                }

                        .hubpage .hub .section3 .win-container {
                                margin-bottom: 0px;
                                margin-left: 0px;
                                margin-right: 0px;
                        }

                        .hubpage .hub .section3 .hourlyTemplatecontainer {
                                display: -ms-grid;
                                -ms-grid-columns: 33% 32% 14% 15%;
                                font-size: 22px;
                                font-weight: lighter;
                                text-align: center;
                                width: 90%;
                                line-height: 24px;
                                border-bottom: 2px dotted;
                        }

                        #hourlyHourCell {
                                -ms-grid-column: 1;
                                height: 24px;
                                margin: auto 0;
                        }

                        #hourlyIconCell {
                                -ms-grid-column: 2;
                                text-align: center;
                        }

                        #hourlyIconCell img {
                                height: 60px;
                                width: auto;
                        }

                                #hourlyTempIconCell {
                                        -ms-grid-column: 3;
                                        text-align: right;
                                        height: 75px;
                                }

                                #hourlyTempIconCell img {
                                        height: 36px;
                                        margin-top: 18px;

                                }

                                #hourlyTempCell {
                                        -ms-grid-column: 4;
                                        height: 34px;
                                        margin: auto 0;
                                        font-size: 32px;
                                        font-weight: lighter;
                                        line-height: 34px;
                                }

        /* ------------- SECTION 4 ------------------------------------------------------------------------------------------
*/

                .hubpage .hub .section4 {
                        top: 10px;
                        height: 100%;
```

```
        }

        .hubpage .hub .section4 .win-container {

            }

            .hubpage .hub .section4 .locationTemplatecontainer {
                display: -ms-grid;
                -ms-grid-rows: 20px 1fr;
                height: 70px;
                background-image: url('/images/locListBG.png');
                -ms-grid-columns: 1fr;
                width: 93%;
            }

            #LocationImg {
                -ms-grid-row: 1;
                text-align: right;
            }

            #LocationCell {
                text-align: center;
                -ms-grid-row: 2;
                font-size: 24px;
            }

            .win-autosuggestbox-hithighlight-span {
                font-size: 22px;
            }

        .win-autosuggestbox {
            width: 86%;
            margin-left: 2%;
        }

        .win-textbox {
            font-size: 22px;
        }

        .hubpage .hub .section {
            margin-left: 20px;
            margin-right: 0px;
        }

        .listView {
            height:72vh;
        }

.label-extralarge {
    font-size: 28pt;
    line-height: 30pt;
}

.label-medium {
    font-size: 15pt;
    line-height: 17.5pt;
}

.label-mediummid {
    font-size: 15pt;
    line-height: 17.5pt;
}
```

section1Page.html

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title></title>
  <script src="/pages/hub/section1Page.js"></script>
</head>
<body>
  <div class="fragment section1page">
    <section aria-label="Main content" role="main">

      <div id="current_data1">
        <div id="fullLocation" class="title label-extralarge" data-win-res="{ textContent: 'location1'
}"></div>
        <div id="current_data1b">
          <div id="current_data1c">
            <div id="curWeather" class="subtext" data-win-res="{ textContent: 'description1'
}"></div>
            <div id="curTemp">?</div>
          </div>
          <div id="current_data1d">
            <img src="/images/wxugLogo.png" id="curWeatherIcon" />
          </div>
        </div>
      </div>

      <div id="todayTable">
        <div id="current_data2">
          <table id="s1_t1">
            <tbody>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'feelslike1' }"></td>
                <td id="curFeelsLike" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'humidity1' }"></td>
                <td id="curHumidity" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'visibility1' }"></td>
                <td id="curVisibility" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'winddirection1' }"></td>
                <td id="curWindDir" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'windspeed1' }"></td>
                <td id="curWindSpeed" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'barometer1' }"></td>
                <td id="curBarometer" class="vtd"></td>
              </tr>
              <tr class="s1_t1_r">
                <td class="s1_t1_td" data-win-res="{ textContent: 'uvindex1' }"></td>
                <td id="curUV" class="vtd"></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>

    </section>
  </div>
</body>
</html>
```

section2Page.html

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="/pages/hub/section2Page.js"></script>
</head>
<body>
  <div class="fragment section2page">
    <section class="sectioncontrol" aria-label="Main content" role="main">

      <!-- Daily ListView-->
      <div class="dailyItemTemplate" data-win-control="WinJS.Binding.Template" style="display:
none">
        <div class="dailyTemplatecontainer">
          <div id="dailyCell" data-win-bind="textContent: day"></div>
          <div id="weekDay" data-win-bind="textContent: weekday"></div>
          <div id="dailyIconCell"><img src="#" class="dailyIcon" data-win-bind="src: dailyIcon"
/></div>
          <div id="dailyTempIconCell"><img src="#" class="dailyListTempIcon" data-win-
bind="src: dailyTempIcon" /></div>
          <div id="dailyMinMaxTemp">
            <span data-win-bind="textContent: maxTemp"></span><br />
            <span data-win-bind="textContent: minTemp"></span>
          </div>
          <div id="dailyHumiIconCell"><img src="/images/humiIcon.png"
class="dailyListHumiIcon" /></div>
          <div id="dailyHumi" data-win-bind="textContent: humi"></div>
        </div>
      </div>

      <!-- The declarative markup necesary for ListView instantiation -->
      <!-- Call WinJS.UI.processAll() in your initialization code -->
      <div class="listView win-selectionstylefilled"
          data-win-control="WinJS.UI.ListView"
          data-win-options="{
        itemDataSource: Daily.ListView.data.dataSource,
        itemTemplate: select('.dailyItemTemplate'),
        selectionMode: WinJS.UI.SelectionMode.none,
        tapBehavior: WinJS.UI.TapBehavior.none,
        layout: { type: WinJS.UI.ListLayout }
        }">
      </div>

    </section>
  </div>
</body>
</html>
```

section3Page.html

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <script src="/pages/hub/section3Page.js"></script>
</head>
<body>
  <div class="fragment section3page">
    <section aria-label="Main content" role="main">

      <div class="hourlyItemTemplate" data-win-control="WinJS.Binding.Template" style="display:
none">
        <div class="hourlyTemplatecontainer">
```

```
                        <div id="hourlyHourCell" data-win-bind="textContent: hour"></div>
                        <div id="hourlyIconCell"><img src="#" data-win-bind="src: hourlyicon" /></div>
                        <div id="hourlyTempIconCell"><img src="#" data-win-bind="src: hourTempIcon"
/></div>
                        <div id="hourlyTempCell" data-win-bind="textContent: hourTemp"></div>
                    </div>
                </div>

                <!-- The declarative markup necesary for ListView instantiation -->
                <!-- Call WinJS.UI.processAll() in your initialization code -->
                <div class="listView win-selectionstylefilled"
                    data-win-control="WinJS.UI.ListView"
                    data-win-options="{
                      itemDataSource: Hourly.ListView.data.dataSource,
                      itemTemplate: select('.hourlyItemTemplate'),
                      selectionMode: WinJS.UI.SelectionMode.none,
                      tapBehavior: WinJS.UI.TapBehavior.none,
                      layout: { type: WinJS.UI.ListLayout }
                      }">
                </div>

            </section>
        </div>
    </body>
    </html>
```

section4Page.html

```
    <!DOCTYPE html>
    <html>
    <head>
      <title></title>
      <script src="/pages/hub/hub.js"></script>
    </head>
    <body>

      <div class="fragment section4page">
        <section aria-label="Main content" role="main">

          <div id="searchBox" class="box" style="display: none">
            <div data-win-control="WinJS.UI.AutoSuggestBox" data-win-options="{
                placeholderText: 'Type a city',
                onsuggestionsrequested: LocationAutoSearchBox.suggestionsRequestedHandler,
                onquerysubmitted: LocationAutoSearchBox.querySubmittedHandler
              }"></div>
            <div id="status"></div>
          </div>

          <div class="locationItemTemplate" data-win-control="WinJS.Binding.Template">
            <div class="locationTemplatecontainer">
              <div id="LocationCell" data-win-bind="textContent: name"></div>
              <div id="LocationImg"><img src="#" class="homeiIcon" data-win-bind="src: img"
/></div>
            </div>
          </div>
          <div class="itemslist win-selectionstylefilled" data-win-control="WinJS.UI.ListView" data-win-
options="{
              selectionMode: 'single',
              tapBehavior: 'directSelect',
              itemTemplate: select('.locationItemTemplate'),
              footer: select('.footer')
            }">
          </div>

          <div class="footer searchbtn">
            <button id="openSearch" class="win-button">add location</button>
```

```
                </div>

            </section>
        </div>

    </body>
    </html>
```

section4Page.js

```
    (function () {
        "use strict";

        var ControlConstructor = WinJS.UI.Pages.define("/pages/hub/section4Page.html", {
            // This function is called after the page control contents
            // have been loaded, controls have been activated, and
            // the resulting elements have been parented to the DOM.
            ready: function (element, options) {
                options = options || {};

                var listView = element.querySelector(".itemslist").winControl;

                listView.itemDataSource = options.dataSource;
                listView.layout = options.layout;
                listView.oniteminvoked = options.oniteminvoked;

                document.querySelector("#openSearch").addEventListener("click", openSearch, false);

            }
        });

        // The following lines expose this control constructor as a global.
        // This lets you use the control as a declarative control inside the
        // data-win-control attribute.

        WinJS.Namespace.define("HubApps_SectionControls", {
            Section4Control: ControlConstructor
        });

        function openSearch() {
            searchBox.style.display = "block";
        }
    })();
```

settings.html

```
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="utf-8" />
        <title>settings</title>

        <link href="/settings/settings.css" rel="stylesheet" />
        <script src="/pages/hub/hub.js"></script>
    </head>
    <body>
        <div class="settings fragment">
            <header class="page-header" aria-label="Header content" role="banner">
                <button class="back-button" data-win-control="WinJS.UI.BackButton"></button>
                <h1 class="titlearea win-type-ellipsis">
                    <span id="settingsPageTitle" class="pagetitle" data-win-res="{ textContent: 'sHeader'
}"></span>
                </h1>
            </header>
            <section class="page-section" aria-label="Main content" role="main">
```

```
                    <div id="settingsBox">

                        <div id="generalSettings" class="headerA" data-win-res="{ textContent: 'sGeneral' }"></div>

                        <div id="unitSection">
                            <div id="units" class="headerB" data-win-res="{ textContent: 'sUnits' }"></div>
                            <div id="setC"><label id="forCels" for="C" data-win-res="{ textContent: 'CLabel'
}"></label><input type="radio" name="unitsVar" value="Celsius" id="Celsius" /></div>
                            <div id="setF"><label id="forFah" for="F" data-win-res="{ textContent: 'FLabel'
}"></label><input type="radio" name="unitsVar" value="Fahrenheit" id="Fahrenheit" /></div>
                        </div>

                        <div id="launchLocationSection">
                            <div id="launchLocation" class="headerB" data-win-res="{ textContent:
'sLaunch_Location' }"></div>
                            <div id="setCurrent"><label id="setCur" for="CurrentLoc" data-win-res="{ textContent:
'CurrentLabel' }"></label><input type="radio" name="LaunchLoc" value="CurrentLocation"
id="CurrentLocation" /></div>
                            <div id="setHome"><label id="setHom" for="HomeLoc" data-win-res="{ textContent:
'HomeLabel' }"></label><input type="radio" name="LaunchLoc" value="HomeLocation" id="HomeLocation"
/></div>
                        </div>



                    </div>
                </section>
            </div>
        </body>
        </html>
```

settings.css

```
        #informationMessage {
                width: 100%;
                height: 100%;
        }

        #settingsPageTitle {
                font-weight: lighter;
                font-size: 26px;
                position: relative;
                height: 42px;
        }

        #settingsBox {
                margin-left: 10px;
                margin-top: 16px;
        }

        .headerA {
                font-size: 20px;
                font-weight: lighter;
                text-transform: uppercase;
                margin-bottom: 10px;
                color: #D4D4D4;
                background-image: url('/images/locListBG.png');
        }

        #unitSection {
                font-size: 26px;
                font-weight: lighter;
                margin-bottom: 16px;
        }
```

```
#units {
	background-image: url('/images/listBG.png');
}

#setC {
	display: -ms-grid;
	-ms-grid-columns: 60% 1fr;
	margin-top: 6px;
	margin-bottom: 12px;
}

#forCels {
	-ms-grid-column: 1;
	-ms-grid-row: 1;
	height: 30px;
	display: block;
	margin: auto 0;
}

#Celsius {
	-ms-grid-column: 2;
	-ms-grid-row: 1;
	margin-right: 100px;
	height: 32px;
}

#setF {
	display: -ms-grid;
	-ms-grid-columns: 60% 1fr;
	margin-top: 6px;
	margin-bottom: 12px;
}

#forFah {
	-ms-grid-column: 1;
	-ms-grid-row: 1;
	height: 30px;
	display: block;
	margin: auto 0;
}

#Fahrenheit {
	-ms-grid-column: 2;
	-ms-grid-row: 1;
	margin-right: 100px;
	height: 32px;
}

#launchLocationSection {
	font-size: 26px;
	font-weight: lighter;
	margin-bottom: 16px;
}

#launchLocation {
	background-image: url('/images/listBG.png');
}

#setCurrent {
	display: -ms-grid;
	-ms-grid-columns: 60% 1fr;
	margin-top: 6px;
	margin-bottom: 12px;
}

#setCur {
	-ms-grid-column: 1;
```

```
        -ms-grid-row: 1;
        height: 30px;
        display: block;
        margin: auto 0;
}

#CurrentLocation {
        -ms-grid-column: 2;
        -ms-grid-row: 1;
        margin-right: 100px;
        height: 32px;
}

#setHome {
        display: -ms-grid;
        -ms-grid-columns: 60% 1fr;
        margin-top: 6px;
        margin-bottom: 20px;
}

#setHom {
        -ms-grid-column: 1;
        -ms-grid-row: 1;
        height: 30px;
        display: block;
        margin: auto 0;
        }

#HomeLocation {
        -ms-grid-column: 2;
        -ms-grid-row: 1;
        margin-right: 100px;
        height: 32px;
}

#servicesSection {
        display: -ms-grid;
        -ms-grid-columns: 60% 1fr;
}

#locationServices {
        -ms-grid-column: 1;
        -ms-grid-row: 1;
        height: 30px;
        display: block;
        font-size: 26px;
        font-weight: lighter;
        margin-top: 20px;
}

#locationToggle {
        -ms-grid-column: 2;
        -ms-grid-row: 1;
        margin-bottom: 45px;
        margin-top: 0px;
}
```

about.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>about</title>
   <link href="/settings/about.css" rel="stylesheet" />
</head>
```

```
<body>

    <div class="aboutBox">
        <div id="about1" class="titleabout">
            <div id="topTitle" class="aboutTitle">Informations</div><img id="aboutIMG"
src="/images/4SeasonsLogo.png" />
        </div>
        <div id="about2">
            <div class="aboutDesc"><strong>Name: </strong>4Seasons</div><br />
            <div class="aboutDesc"><strong>Published by: </strong>Paris Kostopoulos - Giorgos
Antonopoulos</div><br />
            <div class="aboutDesc"><strong>Description: </strong>Weather Application</div><br />
            <div class="aboutDesc"><strong>Mobile OS: </strong>Windows Phone 8.1</div><br /><br />
            <div class="aboutDesc"><strong>Package Name: </strong><br />fdf52aa3-e796-4874-8571-
81eb3cba90a8</div><br />
            <div class="aboutDesc"><strong>Version: </strong>1.0.0.0</div><br />
        </div>
        </div>

    <div class="aboutBox">
        <div id="about1" class="titleabout">
            <div class="aboutTitle">Web Service</div><img id="aboutWU" src="/images/wxugLogo.png"
/>
        </div>
        <div id="about2">
            <div class="aboutDesc"><strong>Name: </strong>Weather Underground</div><br />
            <div class="aboutDesc"><strong>Description: </strong>Data provided by Weather
Underground</div><br />
            <div class="aboutDesc"><strong>API Key ID: </strong>ee4561125223576a</div><br />
        </div>
    </div>

</body>
</html>
```

about.css

```
.aboutTitle {
        font-size: 26px;
        font-weight: lighter;
        margin-top: 0px;
}

#topTitle {
        margin-top: 24px;
}

.aboutDesc {
        font-size: 16px;
        margin-top: -6px;
}

#aboutIMG {
        height: 60px;
        width: 60px;
        margin-top: 14px;
        right: 40px;
        position: fixed;


}
#aboutWU {
        height: 50px;
        width: 50px;
        -ms-grid-column: 2;
        position: fixed;
```

```
		right: 40px;
	}

	.aboutBox {
		display: -ms-grid;
		-ms-grid-rows: 60px 1fr;
		margin-left: 10px;

	}

	#about1 {
		-ms-grid-row: 1;
		display: -ms-grid;

	}

	#about2 {
		-ms-grid-row: 2;
		margin-top: 16px;
	}

	.aboutTitle {
		-ms-grid-column: 1;
	}

	.titleabout {
		-ms-grid-columns: 300px 1fr;
	}

	#aboutIMG {
		-ms-grid-column: 2;
	}

	.titleabout {
	}
```

resources.resjson

```
	// For details on localizing a Windows app, see: http://go.microsoft.com/fwlink/?LinkId=212836

	{
		// Section 1 - Today
	  "Section1": "today",
		"fullLocation": "Location",
		"description": "Description",
		"feelslike1": "Feels like:",
		"humidity1": "Humidity:",
		"uvindex1": "UV Index:",
		"visibility1": "Visibility:",
		"barometer1": "Barometer:",
		"winddirection1": "Wind Dir:",
		"windspeed1": "Wind Speed:",
		// Section 2 - Daily
	  "Section2": "daily",
	  "dayTapDescription": "Select a day for more information",
		// Section 3 - Hourly
	  "Section3": "hourly",
		// Section 4 - Locations
	  "Section4": "locations",
		// Handle Get Location Event
		"gettingLocation": "Getting location...",
		// Handle Call Event
		"gettingData": "Getting data...",
		// Handle Call Error
		"callErrorMessage": "Sorry, we were not able to get weather.<br/>Please check your internet
connection or try again later.",
```

```
                    // Settings Page
                    "sHeader": "Welcome to settings",
                    "sGeneral": "General",
                    "sUnits": "Standard unit",
                    "CLabel": "Celsius",
                    "FLabel": "Fahrenheit",
                    "sLaunch_Location": "Launch location",
                    "CurrentLabel": "Current Location",
                    "HomeLabel": "Home Location",
                    "sServices": "Services",
                    "sLocationServices": "Location Services"
        }
```

package.appxmanifest

```
        <?xml version="1.0" encoding="utf-8"?>
        <Package xmlns="http://schemas.microsoft.com/appx/2010/manifest"
xmlns:m2="http://schemas.microsoft.com/appx/2013/manifest"
xmlns:m3="http://schemas.microsoft.com/appx/2014/manifest"
xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest"
xmlns:build="http://schemas.microsoft.com/developer/appx/2012/build" IgnorableNamespaces="build">
          <Identity Name="fdf52aa3-e796-4874-8571-81eb3cba90a8" Version="1.0.0.0" Publisher="CN=Web"
/>
          <mp:PhoneIdentity PhoneProductId="fdf52aa3-e796-4874-8571-81eb3cba90a8"
PhonePublisherId="00000000-0000-0000-0000-000000000000" />
          <Properties>
            <DisplayName>4 Seasons - Windows Phone 8.1 OS - Weather App</DisplayName>
            <PublisherDisplayName>Paris Kostopoulos - George Antonopoulos</PublisherDisplayName>
            <Logo>images\StoreLogo.png</Logo>
          </Properties>
          <Prerequisites>
            <OSMinVersion>6.3.1</OSMinVersion>
            <OSMaxVersionTested>6.3.1</OSMaxVersionTested>
          </Prerequisites>
          <Resources>
            <Resource Language="x-generate" />
          </Resources>
          <Applications>
            <Application Id="App" StartPage="default.html">
              <m3:VisualElements DisplayName="4 Seasons"
Square150x150Logo="images\Square150x150Logo.png" Square44x44Logo="images\Square44x44Logo.png"
Description="4 Seasons - Windows Phone 8.1 OS - Weather App" ForegroundText="light"
BackgroundColor="#2c287f">
                <m3:DefaultTile Wide310x150Logo="images\Wide310x150Logo.png"
Square71x71Logo="images\Square71x71Logo.png">
                </m3:DefaultTile>
                <m3:SplashScreen Image="images\SplashScreen.png" BackgroundColor="#2c287f" />
                <m3:InitialRotationPreference>
                  <m3:Rotation Preference="portrait" />
                </m3:InitialRotationPreference>
              </m3:VisualElements>
            </Application>
          </Applications>
          <Capabilities>
            <Capability Name="internetClientServer" />
            <DeviceCapability Name="location" />
          </Capabilities>
          <build:Metadata>
            <build:Item Name="Blend" Version="12.0.51020.0" />
          </build:Metadata>
        </Package>
```

saveState.json

```
        {
```

```
    "LOCATIONS": [
       { "name": "Athens, Greece", "home": true , "img": "/images/homeflag.png" }
    ],

    "SETTINGS": { "unitsVariable": "Celsius", "launchLocationVariable": "HomeLocation" },

    "DATA":  {

       }
}
```