7. Ανάπτυξη Εφαρμογής

7.1 Βάση δεδομένων

Για την αποθήκευση και διαχείριση των δεδομένων επιλέχθηκε το εργαλείο διαχείρισης βάσεων δεδομένων SQL Server. Συγκεκριμένα για την διεπαφή του χρήστη με τα δεδομένα και τις οντότητες της βάσης χρησιμοποιήθηκε το SQL Server Management Studio 2017.

7.2 Δεδομένα

Τα δεδομένα που απαιτήθηκαν για την υλοποίηση της εφαρμογής προέρχονται από 2 πηγές

7.2.1 Ergast Api

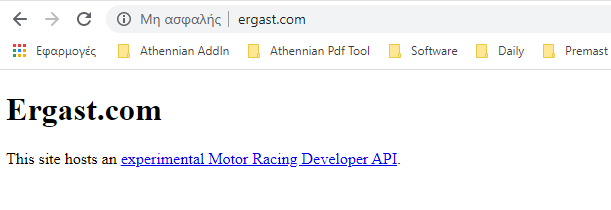


Figure erg - ergast.com

Ο συγκεκριμένος δικτυακός τόπος προσφέρει αρκετά δεδομένα από αγώνες παρελθόντων ετών σχετικά με τον μηχανοκίνητο αθλητισμό. Μάλιστα τα προσφέρει είτε «δυναμικά», αντλώντας τα μέσω http requests σε συγκεκριμένα endpoints ανάλογα με το είδος δεδομένων που απιτείται κάθε φορά, είτε «στατικά». Στην δεύτερη περίπτωση ο χρήστης πρέπει να «κατεβάσει» μια λίστα από sql scripts και να τα τρέξει σε μια βάση δεδομένων που θα δημιουργήσει ώστε να χτίσει τις οντότητες και στην συνέχεια να προσθέσει δεδομένα σε αυτές πάλι εκτελώντας sql scripts. Αυτός είναι και ο τρόπος που επιλέχθηκε στην παρούσα περίπτωση, καθώς παρατηρήθηκε μια καθυστέρηση στην απόκριση των requests γεγονός που καθιστούσε προβληματική την αποδοτική διαχείρισή τους και επίσης όχι και τόσο φιλική προς τον χρήστη.

7.2.1.1 Sql Scripts

Παρακάτω φαίνεται η λίστα με τα scripts που απαιτούνται για την δημιουργία της βάσης δεδομένων. Η ονοματολογία είναι προφανής.

Κάθε script ξεκινά με **έναν αριθμό** που καθορίζει και την σειρά εκτέλεσης. Στην συνέχεια ακολουθεί μια λέξη κλειδί η οποία φανερώνει την ενέργεια

* **CreateTable** : Δημιουργία πίνακα
* **UpdateTable**: Ενημέρωση της δομής του πίνακα π.χ. προσθήκη ενός Primary ή Foreign key
* **InsertInto** : Προσθήκη δεδομένων σε πίνακα

Τέλος ακολουθεί **το όνομα του πίνακα** που επεξεργαζόμαστε κάθε φορά.



Figure - δομή ονόματος sql script

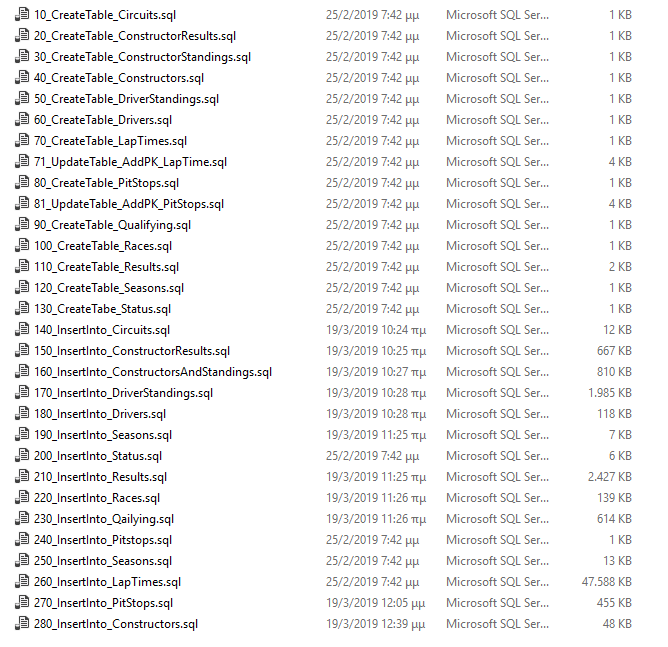
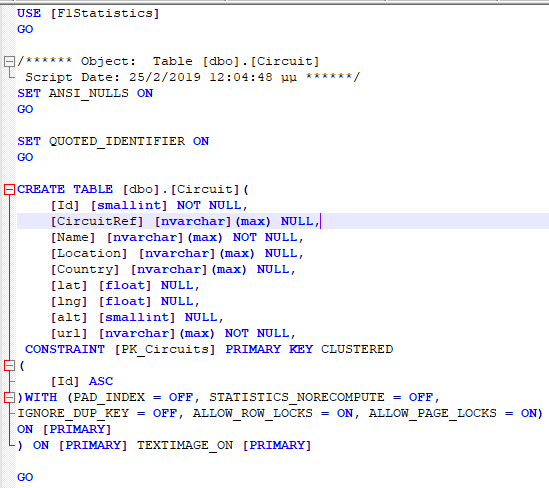


Figure 3 - sql scripts

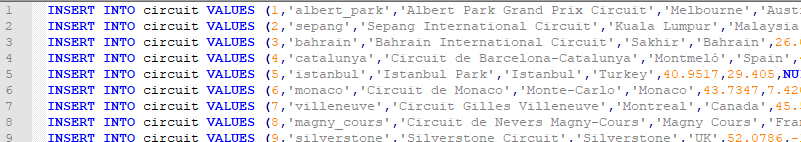
7.2.1.2 Περιγραφή sql script

* CreateTable\_XXXXX

Το παρακάτω script δημιουργεί τον πίνακα “Circuit” και θέτει σαν Primary key το πεδίο Id



* InsertInto\_XXXXX  
    
  Το παρακάτω script εισάγει δεδομένα στον πίνακα Circuit



7.2.2 Καιρικά δεδομένα μέσω του Dark Sky Api

To Dark Sky Api προσφέρει καιρικά δεδομένα τόσο σε πραγματικό χρόνο όσο και παρελθόντων στιγμών. Οπότε ήταν μια καλή και αξιόπιστη λύση για να προβάλλουμε καιρικά δεδομένα της ημέρας διεξαγωγής του αγώνα, γνωρίζοντας ότι ο καιρός πάιζει καθοριστικό ρόλο στην διεξαγωγή και μερικές φορές μάλιστα καθορίζει και την έκβαση ενός αγώνα.

7.2.2.1 Εγγραφή στην υπηρεσία

Για να χρησιμοποιήσει ένας χρήστης το Dark Sky Api αρκεί να ολοκληρώσει μια διαδικασία εγγραφής χρησιμοποιώντας email Και έναν κωδικό πρόσβασης. Με αυτόν τον τρόπο μπορεί να χρησιμοποιήσει δωρεάν τις δυνατότητες του Api μέχρι το όριο των 1000 requests (3) ανά ημέρα. Μετά την επιτυχημένη εγγραφή ο χρήστης αποκτά ένα μοναδικό μυστικό κωδικό (1) τον οποίο προσαρτά σε κάθε request που κάνει γράφοντας το μέσα στο url string

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Στην παραπάνω εικόνα φαίνεται η οθόνη μετά την επιτυχημένη εγγραφή ενός χρήστη.

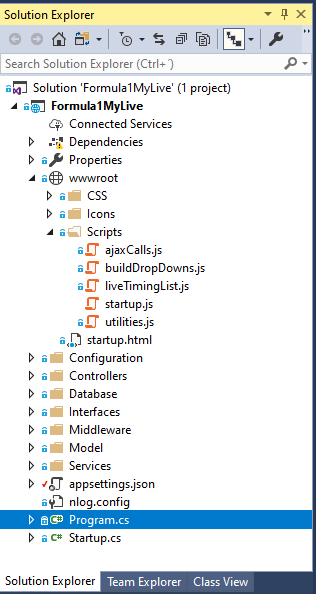
1. Ο μυστικός κωδικός
2. Το ημερήσιο όριο των 1000 call / day
3. ¨Ένα χαρακτηριστικό request της πιο απλής μορφής.

<https://api.darksky.net/forecast/19a7a772f217cfff62a2ΧΧΧΧΧΧΧΧΧΧΧ/37.8267,-122.4233>

Στο τελευταίο κομμάτι του url φαίνονται οι συντεταγμένες του σημείου για το οποίο ζητάμε πληροφορίες.

7.3 Κώδικας της εφαρμογής

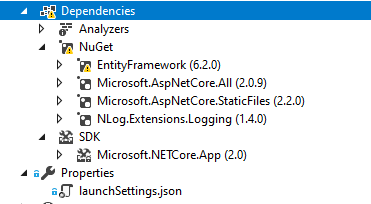
Για την συγγραφή του κώδικα τόσο για το backend όσο και για το frontend χρησιμοποιήθηκε το Visual Studio 2017 Community Edition. Στην παρακάτω εικόνα φαίνεται ένα στιγμιότυπο του Solution Explorer μέσα από το Visual Studio 2017 όπου μπορούμε να δούμε συνοπτικά τους φακέλους και κάποια από τα αρχεία.



Εικόνα 1- VS Solution Explorer

7.3.1 Δομή φακέλων και αρχείων

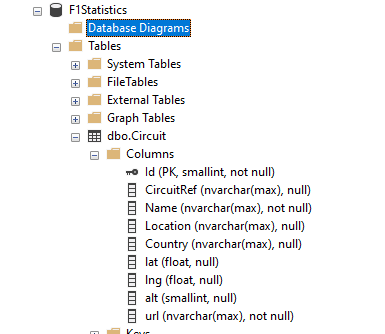
7.3.1.1 Εξαρτήσεις : Nuget packages , SDK & Properties

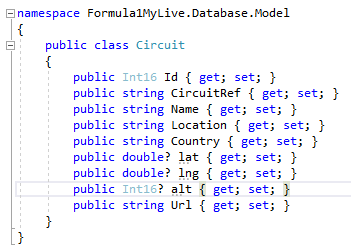


Εικόνα Dependencies, SDK, Properties

**Nuget Packages**  
O «μηχανισμός» Nuget είναι ο προτεινόμενος από την Microsoft τρόπος ώστε οι εφαρμογές που δημιουργούνται να μπορούν να μοιράζονται κοινά κομμάτια κώδικα. Ένα Nuget Package αρχείο είναι κατά βάση ένα συμπιεσμένο αρχείο με επέκταση .nupkg το οποίο περιέχει μεταγλωτισμένο κώδικα (DLL file) και κάποια άλλα απαραίτητα αρχεία συμπεριλαμβανομένου και ενός Manifest αρχείου που περιγράφει το Nuget Package αρχείο όπως για παράδειγμα τον αριθμό της έκδοσης στην οποία αναφέρεται το πακέτο. Αυτά τα πακέτα ο χρήστης μπορεί να τα μεταφορτώσει μέσω του VS να τα εγκαταστήσει μέσω αυτοματοποιημένης διαδικασίας και αν θέλει να τα ενημερώσει σε μεταγενέστερη έκδοση τους όταν αυτή γίνει διαθέσιμή.

* **Entity Framework 6.2.0**  
  Για την αντιστοίχιση των πινάκων / οντοτήτων της βάσης σε backend κλάσεις C# χρησιμοποιήθηκε το κλασικό ORM tool της Microsoft to Entity Framework (EF) . Για παράδειγμα ας επικεντρωθούμε στον πίνακα Circuit. Στην βάση δεδομένων υπάρχει ο πίνακας Circuit o οποίος αντιστοιχίζεται στην κλάση Circuit του μοντέλου όπως φαίνεται και στο παρακάτω σχήμα. Σε επόμενο κεφάλαιο περιγράφεται αναλυτικά η λειτουργία του EF, πως δημιουργούμε την σύνδεση με την βάση δεδομένων και πως ανακτούμε τα δεδομένα.





* **Microsoft.AspNetCore.All**  
  Πρόκειται για ένα μετα-πακέτο δεδομένων το οποίο προσδιορίζει βασικές λειτουργίες μια .Net Core εφαρμογής. Κυρίως παρέχει ένα σύνολο από APIs για να χτιστεί μια Net Core εφαρμογή αλλά και APIs «τρίτων» εφαρμογών όπως του EF
* **Microsoft.AspNetCore.StaticFiles**

Πρόκειται για ένα «Middleware» (έννοια που θα αναπτυχθεί σε επόμενο κεφάλαιο) και χρησιμοποιείται ώστε να μπορεί μια .Net Core εφαρμογής να «σερβίρει» στατικά αρχεία όπως html σελίδες, javascript κώδικα και CSS αρχεία. Συγκεκριμένα στην παρούσα εφαρμογή, όλο το περιεχόμενο που «σερβίρεται» βρίσκεται μέσα στον φάκελο «WWWRoot»

* **NLog.Extensions.Logging**

Η NLοg είναι μια από τις πιο γνωστές πλατφόρμες για την διαχείριση ημερολογίων συμβάντων κοινώς «logs» σε εφαρμογές που αναπτύσσονται σε .Net περιβάλλον. Μας επιτρέπει να «γράφουμε» συμβάντα σε πολλαπλούς προορισμούς όπως σε βάση δεδομένων, σε αρχεία στον τοπικό μας δίσκο ή απλά να τα προβάλουμε στην κονσόλα.

**Software Development Kit (SDK)**

Πρόκειται για ένα σύνολο από βιβλιοθήκες και εργαλεία που επιτρέπει στους προγραμματιστές να δημιουργούμε εφαρμογές που «τρέχουν» σε περιβάλλον .Net Core. Περιλαμβάνει τα παρακάτω τρία βασικά πακέτα:

* Το .NET Core Command Line Interface (CLI)  
  Πρόκειται για ένα σύνολο εργαλείων που δρουν σε μια αλυσίδα/συνέχεια (toolchain) που επιτρέπει την ανάπτυξη, την λειτουργία και την διανομή .net core εφαρμογών σε όλες τις πλατφόρμες και σε πολλαπλά λειτουργικά συστήματα Windows, Linux, MacOS
* .NET Core βιβλιοθήκες

Πρόκειται για το βασικό σύνολο κώδικα που χρησιμεύει στην ανάπτυξη .net core εφαρμογών

* Dotnet driver

Έχει δύο βασικές αρμοδιότητες

* + Να εκτελεί μια .net core εφαρμογή π.χ. dotnet /path/to/my\_app.dll
  + Να εκτελεί μια μεμονωμένη εντολή π.χ. dotnet build

**Properties > launhSettings.json**

Περιλαμβάνει βασικές ρυθμίσεις / παραμέτρους της εφαρμογής ανά προφίλ σε μορφή json. Οι ρυθμίσεις αυτές διαβάζονται από την εφαρμογή κατά την εκκίνηση.



* Όνομα προφίλ: **Formula1MyLive**
* **launchBrowser** : Αν θα εκκινεί και ο browser συνήθως Google Chrome κατά την έναρξη
* Η βασική σελίδα μέσω της οποίας θα σερβίρεται η εφαρμογή: **launchUrl**
* Μεταβλητή περιβάλλοντος : **Development**. Δηλαδή αυτές είναι ρυθμίσεις που ισχύουν κατά την ανάπτυξη της εφαρμογής
* Το domain και το port στο οποίο θα σερβίρεται η εφαμρογή: <http://localhost:62136/>