ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Προχωρημένα Θέματα Τεχνολογίας και Εφαρμογών Βάσεων Δεδομένων

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ για το ακαδημαϊκο έτος 2020-2021

ΟΜΑΔΑ ΝΤΚ

Φοιτητής

Νικολαοσ Τσελικασ-κουρτιδησ, 1780

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΑΪΟΣ 2016

ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

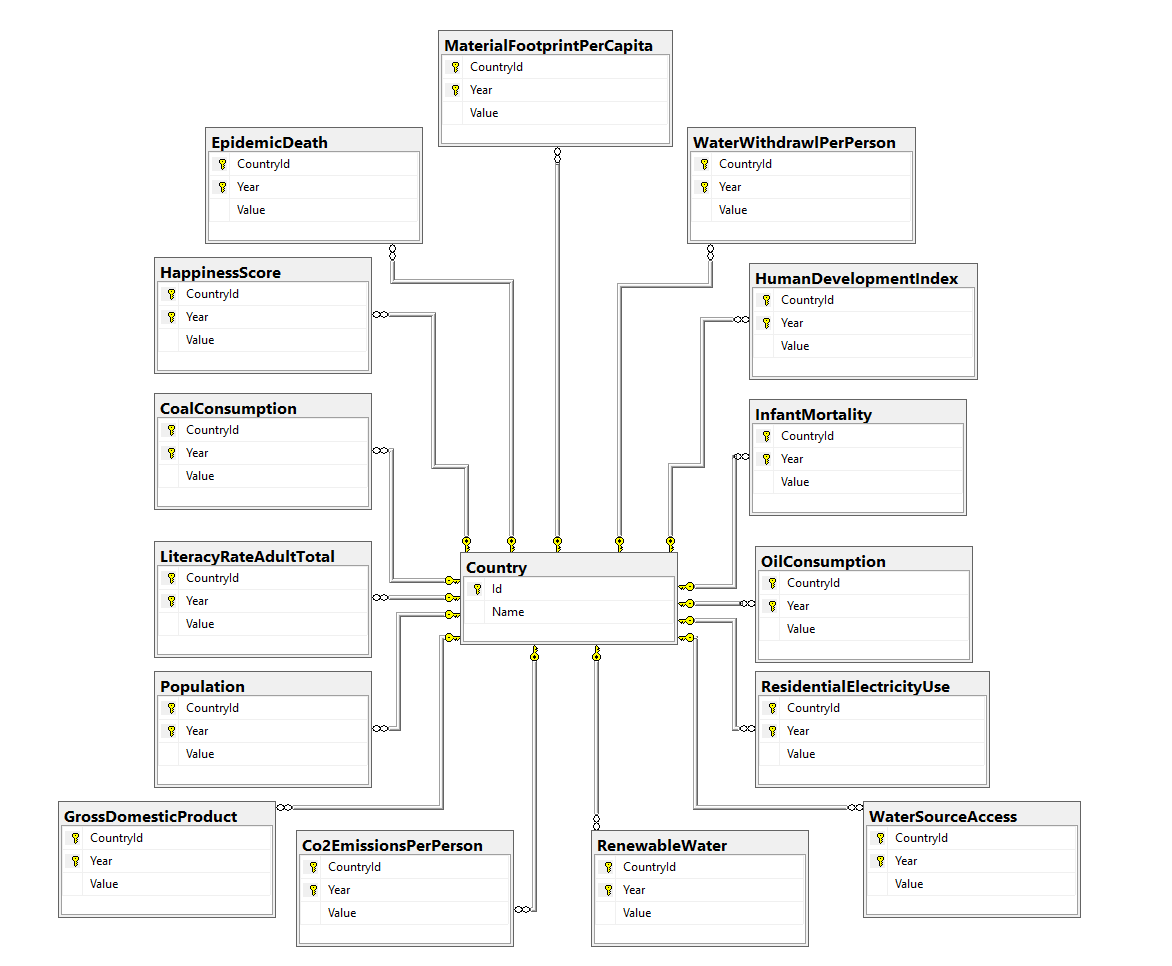
Το ιστορικό των εκδόσεων υπάρχει στο repository, μέσω των commits. Δεν υπήρχε κάποιο συγκεκριμένο roadmap για τις εκδόσεις.

# βαση δεδομενων

Η βάση δεδομένων που χρησιμοποιήθηκε είναι ο SQL Server 2019 της Microsoft (T-SQL).

## Σχεσιακό σχήμα σε λογικό επίπεδο

Το σχεσιακό σχήμα όπως δημιουργήθηκε από το SQL Server Management Studio



Για κάθε δείκτη υπάρχει διαφορετικός πίνακας, ο οποίος έχει σαν Primary Key τα columns CountryId και Year, καθώς θέλουμε ο συνδυασμός αυτό να είναι μοναδικός. Επίσης το CountryId είναι και Foreign Key στον πίνακα Country, ο οποίος έχει πληροφορίες για μία χώρα.

## Σχεσιακό σχήμα σε φυσικο επίπεδο

### Ρύθμιση των παραμέτρων του dbms

Οι ρυθμίσεις του DBMS ήταν στις default. Αυτό σημαίνει πως οι απαιτήσεις της μνήμης αλλάζουν δυναμικά βάση των διαθέσιμων πόρων του συστήματος. Στην περίπτωση μας, μπορεί να φορτώνεται στη μνήμη όλη βάση.

### Ρύθμιση του φυσικού σχήματος της βάσης δεδομένων

Η υλοποίηση έγινε με του Entity Framework Code 5 (ORM της .ΝΕΤ), μέσω του οποίου έγινε mapping των Entity Models και τους πίνακες της βάσης. Το mapping περιλαμβάνει τα relations. Για το performance των ερωτημάτων, χρησιμοποιούταν η μέθοδος Include() του EF Core, για eager loading, να γίνουν δηλαδή τα αντίστοιχα JOINS μόνο όταν του ζητούταν. Διαφορετικά δεν πραγματοποιούταν κάποιο JOIN.

### Ρύθμιση ασφάλειας

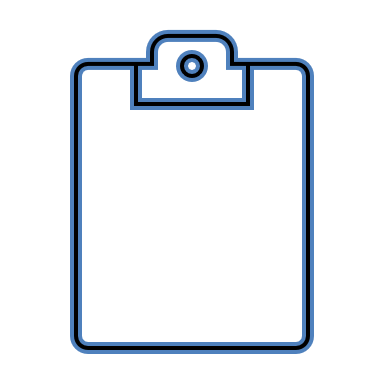
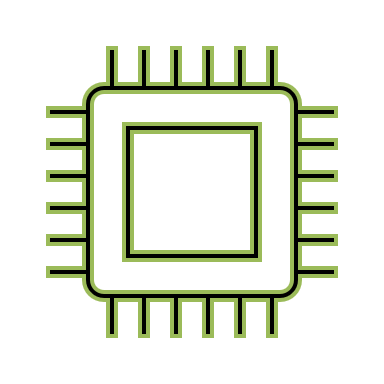
Δεν έγινε κάποια περαιτέρω διαχείριση. Η σύνδεση γινόταν με τον sa user (system admin). Εάν είχαμε περιορισμούς για τα δεδομένα μας, θα έπρεπε να υπάρξει σύστημα διαχείρισης authentication ανά ρόλο. Δηλαδή ο κάθε user της εφαρμογής θα είχε ρόλο, σύμφωνα με τον οποίο θα είχε πρόσβαση στα δεδομένα που του αναλογούσαν.

# αρχιτεκτονικη Λογισμικού

## αρχιτεκτονικη και δομη ETL

Η εισαγωγή δεδομένων γίνεται με τα εξής βήματα:

1. Αποστολή του .csv αρχείου με HttpRequest σε url τύπου base/api/import/{indexName}
2. O ImportController δέχεται το αρχείο και το μεταφέρει στο αντίστοιχο Service
3. To service με τη βοήθεια του GenericParser (nuget package) μετατρέπει το αρχείο σε DataTable (2-D πίνακας)
4. Έπειτα γίνεται mapping από το DataTable στο EntityModel και μεταφέρεται στο αντίστοιχο repository
5. Στο repository γίνεται save στη βάση.



GenericParser

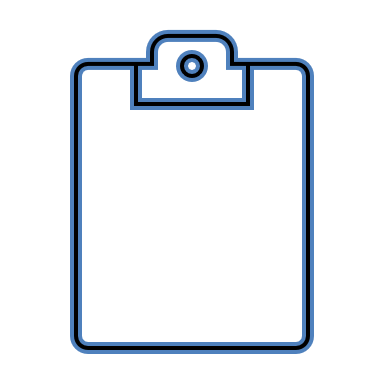
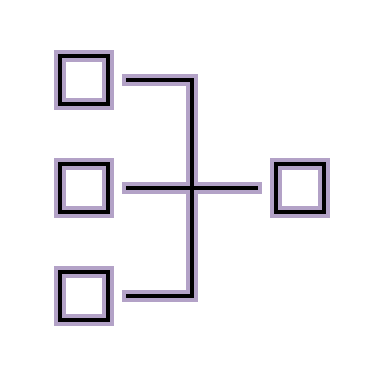
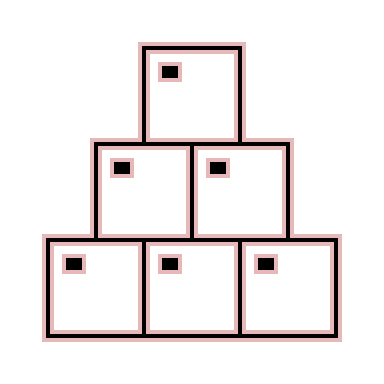
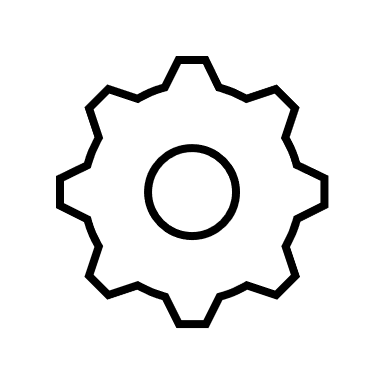
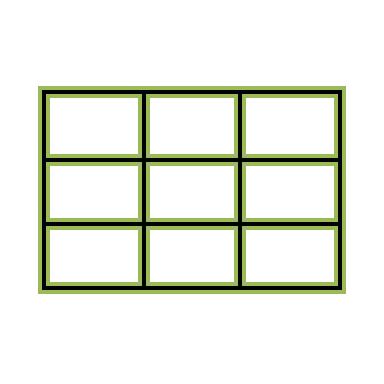
Service

.csv

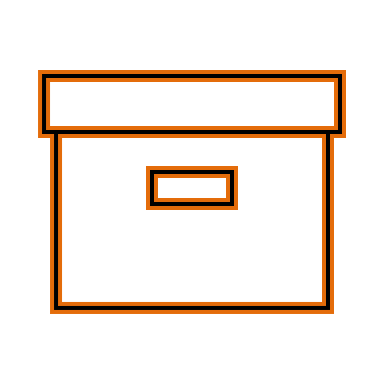
Converts .csv file to DataTable

Database

Controller



Repository

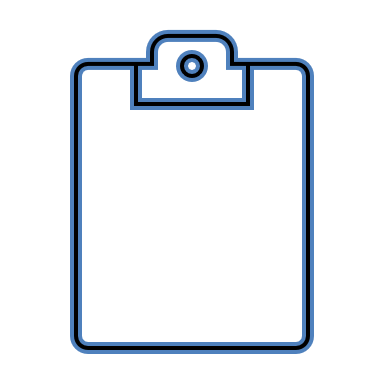


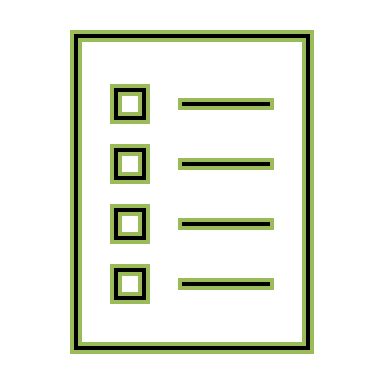
DataTable

.csv

Map DataTable to Entity

Save





Entity

.csv

Για τον GenericParser δεν χρειάζονται κάποιες επιπλέον ρυθμίσεις, καθώς οι default είναι για .csv αρχεία. Να σημειωθεί ότι δεν υποστηρίζεται άλλο είδος αρχείου στην υλοποίηση.

## Διαγράμματα ΠΑΚΕΤΩΝ / υποσυστημάτων kentρικησ εφαρμογησ

Το project αποτελείται από 3 solutions.

1. To database project,
2. Τον server (Web API),
3. Τον client (Blazor WASM)

[1] Το database project έχει όλες τις πληροφορίες για το σχήμα της βάσης. (Tables, Create scripts)

[2] O server είναι ένα web API application. Έχει υλοποιηθεί με 3-Layer αρχιτεκτονική. Συγκεκριμένα έχει το Rest Layer (Controllers, Dtos), το Service Layer (Transformers, Mappings, Any other logic)

[3] O client έχει by default την bootstrap για CSS library και έχει χρησιμοποιηθεί η βιβλιοθήκη ChartJs.Blazor (To repo είναι ChartJs.Blazor.Fork) για την υλοποίηση των διαγραμμάτων.

Τέλος, να σημειωθεί ο τα Entities των Indexes έχουν και 3 πεδία για τον υπολογισμό των πενταετιών, δεκαετιών, εικοσαετιών, ο οποίος γίνεται στο runtime γιατί δεν έχει νόημα να αποθηκευτεί στη βάση αυτή η πληροφορία.

## Διαγράμμα(τα) Κλάσεων κεντρικησ εφαρμογησ

Τα Entity Models είναι σχεδόν ίδια με τους αντίστοιχους πίνακες. Έχουν προστεθεί κάποιες extra λειτουργίες που έχουν να κάνουν με comparison και transiency, όπως επίσης και read-only πεδία για τις πενταετίες, δεκαετίες, εικοσαετίες που υπολογίζονται στο runtime.

Επίσης έχει γίνει χρήση DTOs (Data Transfer Objects) για το output, τα οποία έχουν την πληροφορία που χρειάζεται για κάθε διάγραμμα.

# Υποδείγματα ερωτήσεων και απαντήσεων

Θα καλυφθούν από το video της παρουσίασης, καθώς είναι πολλά τα κομμάτια.

# Τεκμηρίωση και λοιπά σχόλια

Δεν έχει υλοποιηθεί η λειτουργία για να γίνεται φιλτράρισμα βάση πενταετιών, δεκαετιών, εικοσαετιών.