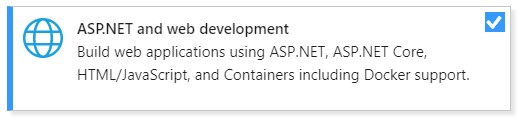
|  |
| --- |
| **DB-Normalization-Tool** |
| **Alex Amanzi - github.com/AAmanzi**  **Matija Luketin - github.com/m-luketin**  **Ante Vuletic - github.com/AnteVuletic** |
| DB-Normalization-Tool je alat za dekompoziciju baze podataka u 3. Normalnu Formu  (3NF) izraden u svrhu seminarskog rada kolegija Baze Podataka 2. |

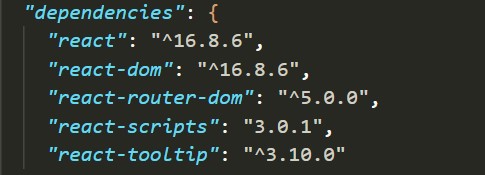
**Ovisnosti:**

Za backend je potrebna instalacija ASP.NET paketa za Visual Studio 2017.

Takoder je potrebno instalirati .NET Core 2.2.107 Software Development Kit.



Za frontend je potrebno instalirati sljedece pakete u pripadajucim verzijama:



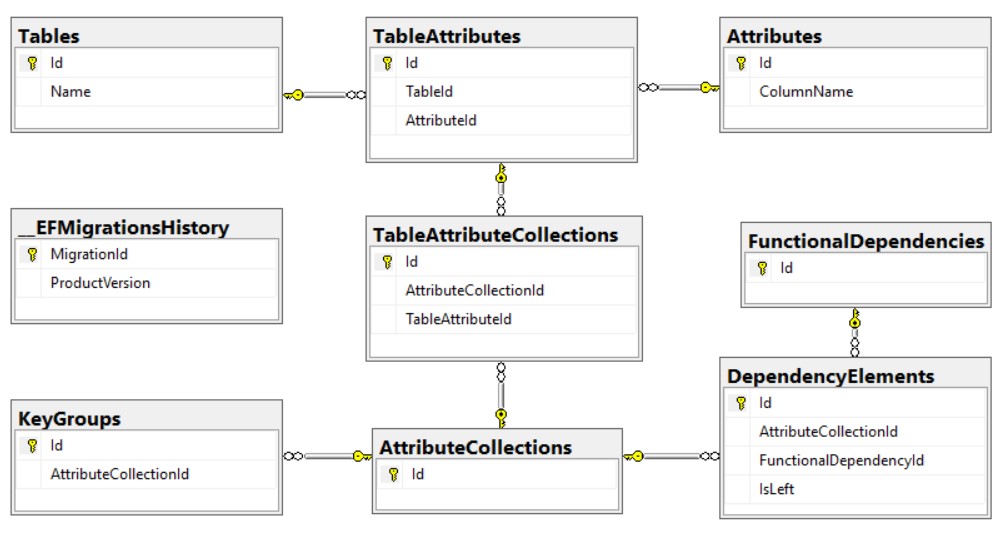
**Pokretanje:**

Backend development server starta se pomocu ISS Express u Visual Studio 2017.

Startanje bez debuggiranja omogucava puno brze navigiranje kroz aplikaciju, ali se ne vide neke funkcionalnosti(npr. Loading screen).

Frontend development server moze se pokrenuti iz komandne linije, navigiranjem u projektni folder te upisivanjem komande ***npm start,*** nakon cega ce se aplikacija automatski otvoriti u default internetskom pregledniku.

**Relacijska shema baze podataka:**

****



public static NormalizedViewModel NormalizeTable(TableViewModel table)

{

var normalizedTable = new NormalizedViewModel {SchemaName = table.Name};

var isAlreadyNormalized = true;

foreach (var dependencyViewModel in table.Dependencies)

{

var isKeyPartOfRelationFrom = table.Keys.Any

(

key => key.All

(partOfKey => dependencyViewModel.From.Any

(dependencyElement => partOfKey == dependencyElement)

)

);

var isRelationToPartOfKey= table.Keys.Any

(

key => key.Any

(partOfKey => dependencyViewModel.To == partOfKey)

);

if (isKeyPartOfRelationFrom || isRelationToPartOfKey) continue;

isAlreadyNormalized = false;

break;

}

if (isAlreadyNormalized)

{

normalizedTable.TableAttributes.Add(table.Attributes);

return normalizedTable;

}

foreach (var dependencyViewModel in table.Dependencies)

{

var decompositionElement = new List<string>();

decompositionElement.AddRange(dependencyViewModel.From);

decompositionElement.Add(dependencyViewModel.To);

if(normalizedTable.TableAttributes.All

(tableAttribute => !decompositionElement.All

(element => tableAttribute.Any(ta => ta.Equals(element)))

)

)

{normalizedTable.TableAttributes.Add(decompositionElement);}

}

if(normalizedTable.TableAttributes.All

(tableAttribute => !table.Keys.First().All

(keyAttribute => tableAttribute.Any(ta => ta.Equals(keyAttribute)))

)

)

{normalizedTable.TableAttributes.Add(table.Keys.First());}

return normalizedTable;

}

**Backend projekti:**

Normalization.Api – zaprima klijentske zahtjeve, te ovisno o njima salje

podatke natrag, ili u nize slojeve aplikacije

Normalization.ViewModel – sluzi za enkapsulaciju podataka u oblik prikladan za frontend

Normalization.Maps – mapiranje modela primljenog s frontenda na nize slojeve, te

takoder sadrzi algoritam, odnosno poslovnu logiku

Normalization.Repository – CRUD implementacija nad Data slojem

Normalization.Data – definira spoj s bazom(kontekst) , te mapira objekte backenda

na entitete spremljene u lokalnoj bazi podataka

**Frontend React komponente:**

MainScreen.js – glavna stranica s opisom projekta

Form – takoder se sastoji od vise komponenti, korisnicko sucelje za unos

podataka potrebnih za izvodenje algoritma

Algorithm.js – opis algoritma

Display – sastoji se od vise komponenti, prikazuje rezultate izvodenja algoritma,

te spremljene tablice

LoadingScreen.js – komponenta koji se prikazuje tijekom ucitavanja podataka

Navbar.js – navigacijski toolbar na vrhu stranice

**Tehnologije**:

* React.js – UI/UX
* .NET Core framework – poslovna logika/algoritam
* EF Core – povezivanje s bazom podataka

**Alati**:

* Visual studio 2017 – C#
* Visual studio Code – HTML, CSS, JS
* Microsoft SQL Server Management Studio – SQL
* Git Bash, GitHub.com – Verzioniranje
* Google Docs, Slack, Discord – Komunikacija