Work Documentation

1. **TASK PROPOSED**

“Web Application com cadastrar, visualizar, deletar e editar vagas de trabalho:  
  
 - Linguagem: .NET (Framework ou Core)  
 - Arquitetura: MVC (ASP.NET MVC)  
 - Armazenamento: Banco de dados  
 - Aparência não é um fator de avaliação”

**Project Evolution**

|  |  |  |
| --- | --- | --- |
| Evolution | Date | Description |
|  | 5/18/2021 | SQL database creation. |
|  | 5/18/2021 | Creation of the Solution project with VS and migration of database |
|  | 5/19/2021 | Failed to continue the development: Problems:   1. Create the Views; 2. Elaborate the logic; and 3. Clean the garbage created by the migration of the database (database first aproach) |
|  |  |  |
|  |  |  |
|  |  |  |

1. **BRIEF ANALISYS**

Analyzing the proposed task, we can realize that it’s a CRUD application for Web, using ASP.NET MVC. So, I decided create a database and then develop a CRUD app (“database first” approach).

1. **DATABASE DESIGN**
   1. Requirements Analysis
      1. In fact, we don’t have much information to support the analysis, so we made a brief study of the Stefanini’s job website. From there I took the information I used.
      2. Due the lack of information, I assumed that there is no need of concern about the job offer approvals and budget and who is requiring to post the job offer.
      3. Otherwise, I assumed that possibly, in future, there would be another App to match the job offers face the job applications. So, some information like “Responsibility” and “Requirements” should be treated individually, making them multivalued attributes.
   2. Initially I observed the reality and defined the Job Offer Entity and its attributes:

* **Tittle**: The name of the Job vacancy. It should be a one or two words.
* **Category**: It’s possible to find in the internet that “Job category means a broad-based group of employees with comparable job responsibilities who are located at comparable levels of responsibility within an organization. I observed categories like: Marketing, Helpdesk, Human Resources, Sales, Consultant, Bank/Financial, Engineering, etc. mentioned in Stefanini’s site.
* **Post** **Date**: Date of the publication of the job offer.
* **Detail**: In this attribute, more details about the job offer are stored. Due the said lack of requirements, I will consider store only plain text.
* **Type**: This attribute should be filled with the type of the job, like: Full-time, part-time, shift, etc.
* **Duration**: Some jobs offer intends to fill a temporary vacancy. If so, this attribute should contain the number of months.
* **Wage**: In my opinion the administration should have a wage control of job offers, for budgeting. I know, otherwise, that this attribute should not be published unless another employer’s intention.
* **Local**: This attribute must contain the information about the local where the employee will work. It’s possible to insert the expression “everywhere” if the job could be done in home office style. So, it needs to be a separate entity.
* **Responsibilities**: This attribute should be filed with the activity that the employee will hold during his activity. Due to the mentioned future match with job seekers, this attribute should be filled by a sufficient number of phrases characteristics that, at the end, will demonstrate the activity characteristics. So, normally, one job offer will have several lines of responsibilities. So, it needs to be a separate entity.
* **Job Requirements**: This attribute should be filed with the employee characteristics that better match with the employer’s necessity. Due to the mentioned future match with job seekers, this attribute should be filled by a sufficient number of phrases characteristics that, at the end, will demonstrate the employers need. So, normally, one job offer will have several lines of requirements. So, it needs to be a separate entity.
  1. These requirements mentioned above lead me to develop the Initial Diagram showed below.

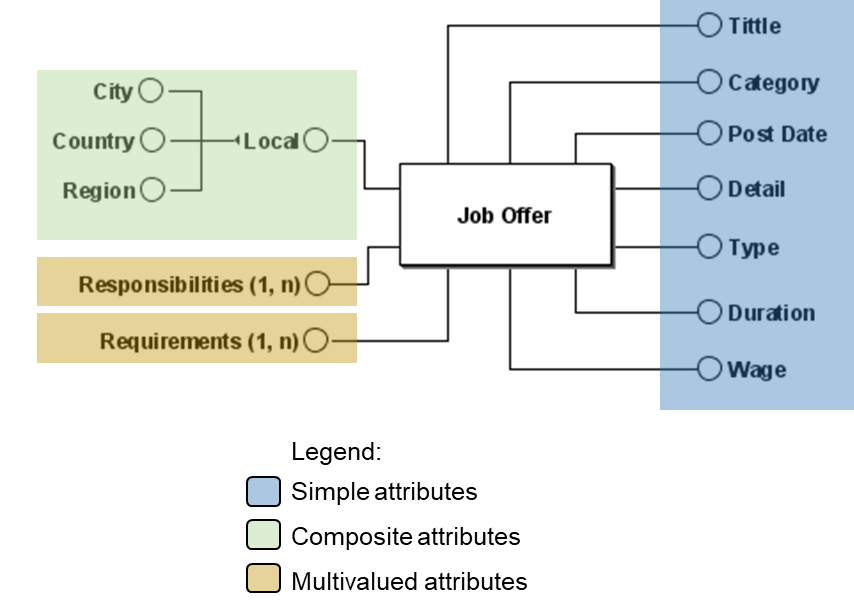


Figure 1 - INITIAL Conceptual Diagram

* 1. Entity-Relationship Model:

Analyzing the diagram and the attributes, I identified the entities listed and the relationship between them:

* + 1. Relevant Entities:
       1. Job Offer
       2. Local
       3. Responsibilities
       4. Requirements
    2. Relationships between these Entities
       1. The Job Offer has one Local (normally); although one Local may have more than one Job Offer;
       2. The Job Offer may have several Responsibilities and many Job Offer may have the same Responsibility; and
       3. The Job Offer may have several Requirements and many Job Offer may have the same Requirement.
    3. To consider further:
       1. If it should be better if to separate the attributes Category and Position Type and consider them as Entities.
    4. Considering the above listed, it was possible to formulate the diagram below.

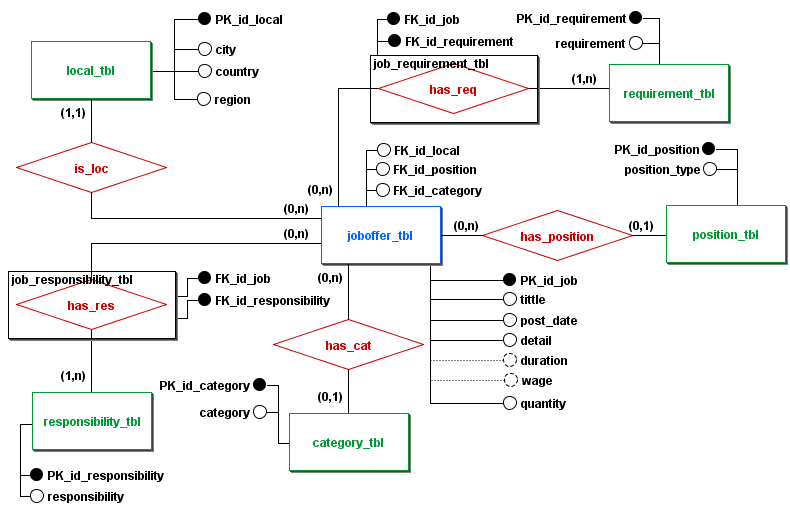


Figure 2 – Final Conceptual Diagram

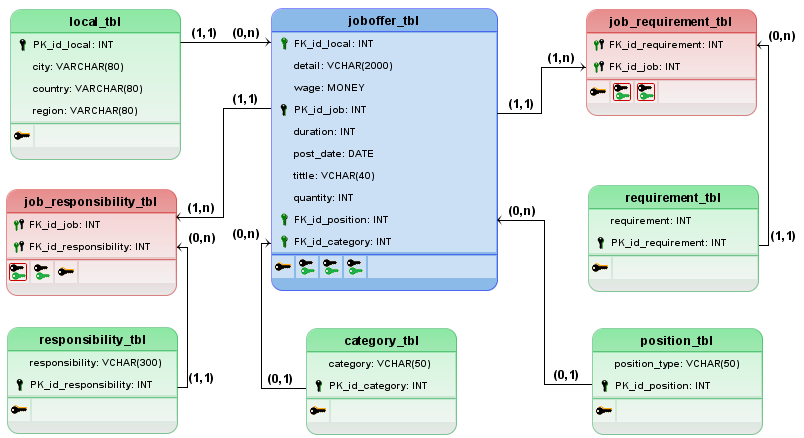


Figure 3 – Final Logic Diagram

* 1. Data Dictionary (Metadata Repository):
     1. Entities and Relationship

In these tables I stablished the final names of the tables and attributes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entities** | | **Relationship** | |  |
| **First Table** | **Second Table** | **Name** | **Cardinality** | **Description of First Tables** |
| joboffer\_tbl | local\_tbl | is | (1,1) | The joboffer\_tbl is the main table of the project. Contains the data information of the job offers. |
| requirement\_tbl | has | (1,n) |
| position\_tbl | has | (0,1) |
| category\_tbl | has | (0,1) |
| responsibility\_tbl | has | (1,n) |
| local\_tbl | joboffer\_tbl | has | (0,n) | The local\_tbl is the table to store the data about the localization of the job vacancy. |
| requirement\_tbl | belongs | (0,n) | The requirement\_tbl contains the requirements that the candidate need to has of the job position. |
| position\_tbl | belongs | (0,n) | The position\_tbl contains the data that indicates how the employee will be contracted. |
| category\_tbl | belongs | (0,n) | The category\_tbl contains the information about the category of the job vacancy. |
| responsibility\_tbl | belongs | (0,n) | The responsibility\_tbl table contains the responsibility held by de employee in the job position. |

* + 1. Attributes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| joboffer\_tbl | PK\_id\_job | INT | 2 | PK  NOT NULL | - | Id number of the job offer. |
| title | VCHAR | 40 | NOT NULL | - | Name of the job offer. |
| post\_date | DATE | 3 | NOT NULL | - | Date of the publication of the offer. |
| detail | VCHAR | 2000 | NOT NULL | - | Details of the job. |
| duration | INT | 2 |  | - | Contains the duration of the job, if is the case, in months. |
| wage | MONEY | 5 |  | - | Contains the amount of the wage for the job. |
| quantity | INT | 2 | NOT NULL | 1 | This attribute contains the quantity of equivalent job vacancies (with identical attributes values). The default value is 1. The user must fill this attribute with the quantity of equivalent job offers. The developer need to build a process to handle with this topic. |
| FK\_id\_local | INT | 2 | FK  NOT NULL | - | Id number of the local to where the job vacancy belongs. |
| FK\_id\_requirement | INT | 2 | FK  NOT NULL | - | Id number of the requirements for a candidate to obtain the position. |
| FK\_id\_type | INT | 2 | FK  NOT NULL | - | Id number of the information about the work regimen (full-time, part-time, shift, contract…). |
| FK\_id\_category | INT | 2 | FK  NOT NULL | - | Id number of the category of the job been offered. |
|  | FK\_id\_responsibility | INT | 2 | NOT NULL | - | Id number of the responsibilities of the employee of the job. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| local\_tbl | PK\_id\_local | INT | 2 | PK | - | Id number of the local of the job vacancy. |
| city | VCHAR | 80 | NOT NULL | - | Name of the city of the job. |
| country | VCHAR | 80 | NOT NULL | - | Name of the country of the job. |
| region | VCHAR | 80 | NOT NULL | - | Name of the region of the job. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| requirement\_tbl | PK\_id\_requirement | INT | 2 | PK | - | Id number of the requirement. |
| requirement | VCHAR | 200 | NOT NULL | - | This row must contain one requirement for the offered job. |
| FK\_id\_job | INT | 2 | FK | - | Identifies the job that belongs the attribute. It is a not unique atributte. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| position\_tbl | PK\_id\_type | INT | 2 | PK | - | Id number of the position type of the job offered. |
| position\_type | VCHAR | 50 | NOT NULL | - | This row should contain the type of the job, like full-time, part-time, shift, trainee, etc.. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| category\_tbl | PK\_id\_category | INT | 2 | PK | - | Id number of the category of the job offered. |
| category | VCHAR | 50 | NOT NULL | - | This row should contain the category of the job offered. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| responsibility\_tbl | PK\_id\_responsibility | INT | 2 | PK | - | Id number of the responsibility of the job offered. |
| responsibility | VCHAR | 300 | NOT NULL | - | This row should contain the responsibility of the job offered. |
| FK\_id\_job | INT | 2 | FK | - | Identifies the job that belongs the attribute. It is a not unique atributte. |

* + 1. Relationships
       1. This table contains the relationships and their informations thas has no problems.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Relationship** | **1st Table** | | **2nd Table** | | **Cardinality** | **Description** |
| **Name** | **Column** | **Name** | **Column** |
| Is | joboffer\_tbl | FK\_id\_local | local\_tbl | PK\_id\_local | (1,1) | This relationship joins the job offer with it’s local. All job offer **must** have **one and only one** local. Otherwise, if the job employee will work from home, it may be put “Remote” or “Anywhere”. |
| has | FK\_id\_position | position\_tbl | PK\_id\_position | (0,1) | This relationship joins the job with its position type. Normally the employer should know the job’s position type. This attribute should be filled with the information, but if it’s unknown, **this attribute may be left empty**. The job must have **no more than one type**. |
| has | FK\_id\_category | category\_tbl | PK\_id\_category | (0,1) | This relationship joins the job with its category. This attribute should be filled with the information, but if it’s unknown, **this attribute may be left empty**. The job must have **no more than one category**. |
| has | local\_tbl | PK\_id\_local | joboffer\_tbl | FK\_id\_local | (0,n) | This relationship joins the local with the jobs existing in their location. Eventually, one local **can have no** vacancy. But one local **can have many** job vacancies. |
| belongs | position\_tbl | PK\_id\_position | FK\_id\_position | (0,n) | This relationship joins the position type with the jobs from which it belongs to. Eventually, one position type **can belong to no** job. But one position type **can belong to many** jobs. |
| belongs | category\_tbl | PK\_id\_category | FK\_id\_category | (0,n) | This relationship joins the category with the jobs from which it belongs to. Eventually, one category **can belong to no** job. But one requirement **can belong to many** jobs. |

* + - 1. This table contains the relationships that need to be treated and two associative entity becose of the (n,m) cases.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Relationship** | **1st Table** | | **2nd Table** | | **Cardinality** | **Description** |
| **Name** | **Column** | **Name** | **Column** |
| has\_req | joboffer\_tbl | FK\_id\_requirement | requirement\_tbl | PK\_id\_requirement | (1,n) | This relationship joins the job with the requirements that the candidates for the job must or should have. The job **must have at least one** requirement (**may have more than one**). |
| has\_res | FK\_id\_responsibility | responsibility\_tbl | PK\_id\_responsibility | (1,n) | This relationship joins the job with its responsibilities. The job **must have at least one** responsibility and **may have more than one**. |
| has\_req | requirement\_tbl | PK\_id\_requirement | joboffer\_tbl | FK\_id\_requirement | (0,n) | This relationship joins the requirement with the jobs from which it belongs to. Eventually, one requirement **can belong to no** job. But one requirement **can belong to many** jobs. |
| has\_res | responsibility\_tbl | PK\_id\_responsibility | FK\_id\_responsibility | (0,n) | This relationship joins the responsibility with the jobs from which it belongs to. Eventually, one responsibility **can belong to no** job. But one responsibility **can belong to many** jobs. |

* + - 1. Associative Tables created:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| job\_responsibility\_tbl | FK\_id\_responsibility | INT | 2 | PK, FK, ON DELETE NO ACTION | - | Id number of the responsibility of the job offered. |
| FK\_id\_job | INT | 2 | PK, FK, ON DELETE SET NULL | - | Identifies the job that belongs the attribute. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Type** | **Length (bytes)** | **Constraint** | **Default** | **Description** |
| job\_requirement\_tbl | FK\_id\_requirement | INT | 2 | PK, FK, ON DELETE NO ACTION | - | Id number of the requirement. |
| FK\_id\_job | INT | 2 | PK, FK, ON DELETE SET NULL | - | Identifies the job that belongs the attribute. |

* + 1. Database normalization
       1. First Normal Form (1NF):

<joboffer\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Primary key | The table has a PK. |
|  | No duplicate tuples | It’s possible that a local have more than one vacancy for one specific job. So, to eliminate this problem, the id\_column was created in each table. Aditionaly, the developer needs to handle with this topic. Probably adding a suffix number in the title name of the job offer, or use a counter attribute. |
|  | No repeating groups | There are no repeating attributes in the designed table. |
|  | Atomic columns (cells have single value) | All cells have single values. |

<local\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Primary key | The table has a PK. This table has a counter as a PK. |
|  | No duplicate tuples | It’s possible that one local has more than one vacancy for one specific job. So, to eliminate this problem, the id\_column was created. And, aditionaly, the attribute city does’t uniquely identify a row, since is possible to have homonimous cities around the world. The developer needs to handle with this topic. Probably adding code to prevent the duplication of a tuple. |
|  | No repeating groups | There are no repeating attributes in the designed table. |
|  | Atomic columns (cells have single value) | All cells have single values. |

<requirement\_tbl> <responsibility\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Primary key | The tables have PK. These tables have a counter as a PK. |
|  | No duplicate tuples | It’s possible that the columns requirement and responsibility have equal values, respectively. But the colums FK\_id\_job will differ and there wil be no duplicate tuples. The developer needs to add code to prevent the duplication of a tuple. |
|  | No repeating groups | There are no repeating attributes in the designed tables. |
|  | Atomic columns (cells have single value) | All cells have single values. |

<has\_req\_tbl> <has\_res\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Primary key | The tables don’t have PK. Thei have only two colums. They don’t need. |
|  | No duplicate tuples | The colums FK\_id\_job will differ and there wil be no duplicate tuples. |
|  | No repeating groups | There are no repeating attributes in the designed tables. |
|  | Atomic columns (cells have single value) | All cells have single values. |

<position\_tbl\_type> <category\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Primary key | The tables have PK. These tables have a counter as a PK. |
|  | No duplicate tuples | It’s not possible that the columns category and position\_type have equal values, respectively. The developer needs to add code to prevent the duplication of a tuple. |
|  | No repeating groups | There are no repeating attributes in the designed tables. |
|  | Atomic columns (cells have single value) | All cells have single values. |

* + - 1. Second Normal Form (2NF)

<joboffer\_tbl> <local\_tbl> <requirement\_tbl> <responsibility\_tbl>   
<position\_tbl\_type> <category\_tbl> <has\_req\_tbl> <has\_res\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Meet 1FN | These tables meet 1FN. |
|  | Every non-prime attribute of the relation is dependent on the whole of every candidate key. | In fact, there is only one prime attribute, the PK, in each one of these tables. All other attributes of the relations are dependent of the respectively PK. |

* + - 1. Third Normal Form (3NF)

<joboffer\_tbl> <requirement\_tbl> <responsibility\_tbl>   
<position\_tbl\_type> <category\_tbl> <has\_req\_tbl> <has\_res\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Meet 1FN | These tables meet 1FN. |
|  | Meet 2FN | These tables meet 2FN. |
|  | There is no transitive dependencies in the relations. | In fact, there is only one prime attribute, the PK, in each one of these tables. All other attributes of the relations are only dependent of the respectively PK. |
|  | All non-prime attributes depend only on the candidate keys | In fact, there is only one prime attribute, the PK, in each one of these tables. All other attributes of the relations are dependent of the respectively PK. |

<local\_tbl>

|  |  |  |
| --- | --- | --- |
| **OK** | **Normal Form Item** | **Notes:** |
|  | Meet 1FN | These tables meet 1FN. |
|  | Meet 2FN | These tables meet 2FN. |
|  | There is no transitive dependencies in the relations. | In fact, there is only one prime attribute, the PK, in each one of these tables. But the other attributes of the relation are dependent of other than the respectively PK. {City} →{Country} and {Country} → {Region}.  The correct normalization in this case is to decompose the table into three more tables. But it’s not worthy. The codes, the UI, -joins, etc. would be hard and it won’t bring advantages. So, I decided to maintain the three attributes together.  This table doesn’t meet the 3FN. |
|  | All non-prime attributes depend only on the candidate keys |

* 1. Naming convention

In this task I intend to follow the convention showed in the next table. Almost all retrived from the Microsoft tutorials.

|  |  |  |
| --- | --- | --- |
| **Items** | **Naming Convention** | **Examples** |
|  |  |  |
| Attribute | PascalCase + Attribute sufix  (it’s not necessary in code – between [ ])  Exceptions:  I will use the under\_line for names of Attributes and the prefixes PK\_, FK\_ and id\_ in some attributes. | // Attribute DllImportAttribute.  [System.Runtime.InteropServices.DllImport("user32.dll")]  extern static void SampleMethod(); |
| Class | PascalCase  Exceptions:  I will use the under\_line for names of Tables and Attributes and the sufix instead of prefix for the tables Job\_Tbl, instead of Tbl\_Job; and  I will use the prefix PK\_, FK\_ and id\_ in some attributes. | public class Manager : Employee  {  // Employee fields, properties, methods and events are inherited  // New Manager fields, properties, methods and events go here...  } |
| Constant | PascalCase | class Calendar2  {  public const int Months = 12, Weeks = 52, Days = 365;  } |
| Enum value | PascalCase | public enum FileMode { Append, ... } |
| Event | PascalCase | public class Process { public event EventHandler Exited; } |
| Field | PascalCase (XMT private - \_camelCase) | public class MessageQueue { public static readonly TimeSpan InfiniteTimeout; } public struct UInt32 { public const Min = 0; } |
| Function | lowerCamelCase | let list1 = [ 1; 2; 3]  let sumPlus x = |
| Identifier | PascalCase | USE AdventureWorks  GO  CREATE TABLE [SalesOrderDetail Table] |
| Interface | I + PascalCase | public interface IEnumerable { ... } |
| Namespace | PascalCase | namespace System.Security { ... } |
| Parameter name | lowerCamelCase | public class Convert { public static int ToInt32(string value); } |
| Private Method | lowerCamelCase |  |
| Private Variable | lowerCamelCase |  |
| Property | PascalCase | public class String { public int Length { get; } } |
| Public member | PascalCase |  |
| Public Method | PascalCase | public class Object { public virtual string ToString(); } |
| Public Variable | CamelCase |  |
| Type | PascalCase | public class StreamReader { ... } |

**<To be continued>**

1. Referencies used:
   1. Jose Carlos Macoratti

video: ASP .NET Core 3.1 com EF Core - DataBase First

site: <https://www.youtube.com/watch?v=Y22WGhUI2rs>

Float, Double e Decimal. Afinal qual usar?

site: <http://www.macoratti.net/12/12/c_num1.htm>

.NET Sugestões e dicas para escever um bom código

site: <http://www.macoratti.net/13/09/net_pcod1.htm>

* 1. Junaid Shahid, from The Engineering Projects, Paquistan

video: Entity Framework Core Database First CRUD Operations in ASP.NET CORE Application

site: <https://www.youtube.com/watch?v=PlW9dgU_aVM>

* 1. Entity Framework Tutorial

site: <https://www.entityframeworktutorial.net/efcore/create-model-for-existing-database-in-ef-core.aspx>

* 1. Connection Strings Tutorial

site: <https://docs.microsoft.com/en-us/ef/core/miscellaneous/connection-strings>

* 1. Reverse Engineering Tutorial

site: <https://docs.microsoft.com/en-us/ef/core/managing-schemas/scaffolding?tabs=dotnet-core-cli>

* 1. Jhonatan de Souza, Dev Aprender.com

Curso SQL Completo 2019 [Iniciantes]+Desafios+Muita Prática,

site: <https://www.youtube.com/watch?v=rX2I7OjLqWE>

* 1. Venkat ([www.PragimTech.com](file:///D:/Projects/Curso%20de%20Modelagem%20de%20Dados/www.PragimTech.com))

Entity framework core seed data

site: <https://www.youtube.com/watch?v=qDUS8ocavBU>

* 1. Fabio dos Reis (Boson Treinamentos http://www.bosontreinamentos.com.br)

Curso de Modelagem de Dados

site:<https://www.youtube.com/watch?v=Q_KTYFgvu1s&list=PLucm8g_ezqNoNHU8tjVeHmRGBFnjDIlxD>

Opções de Chave Estrangeira no MySQL

site: <http://www.bosontreinamentos.com.br/mysql/opcoes-de-chave-estrangeira-mysql/#:~:text=RESTRICT%3A%20Impede%20que%20ocorra%20a,de%20chave%20estrangeira%20%C3%A9%20retornada>.

* 1. Lucidchart

Entity-Relationship-Diagramme

site: <https://www.youtube.com/watch?v=XCkd27GtZoM>

* 1. Professor Drausio

PD – Criar Tabela Utilizando Diagrama (Designer) no SQL Server

site: <https://www.youtube.com/watch?v=PmdKMDaQRhA>

* 1. Guru99

site: <https://www.guru99.com/data-modelling-conceptual-logical.html>

* 1. Science Direct Data Dictionary

site: <https://www.sciencedirect.com/topics/computer-science/data-dictionary>

* 1. Databases - Entity-Relationship Modelling

site: <https://teaching.csse.uwa.edu.au/units/CITS3240/Lectures/db-er1-nup4.pdf>

* 1. US State Procurement Manual

site: <http://vendornet.state.wi.us/vendornet/procman/prod3.pdf>

* 1. Stefanini’s Careers page

site: <https://stefanini.com/en/careers>

* 1. Linguee Dictionary

site: <https://www.linguee.com.br/portugues-ingles/>

* 1. Databasedev.co.uk database solutions

First, Second and Third Normal Form

site: <http://www.databasedev.co.uk>

* 1. Eduka TI

Normalização de Banco de Dados

site: <https://www.youtube.com/channel/UCt8lev8EupaUT9EHtvwYrwg>

Modelagem de banco de dados: Cardinalidade de relacionamento n:m

site: <https://www.youtube.com/watch?v=E-Zu0y8Jbbo>

* 1. stackoverflow

Qual a melhor maneira de se representar um Endereço?

site: <https://pt.stackoverflow.com/questions/622/qual-a-melhor-maneira-de-se-representar-um-endere%C3%A7o>

* 1. Microsoft tutorial
  2. DevMEDIA

Consistência dos Dados e Constraints: SQL Server

site: <https://www.devmedia.com.br/consistencia-dos-dados-e-constraints-sql-server-2005-parte-1/17622>