**Persistence Layer**

In order to persist data in an ASP.NET Web API solution, a database is often required. There are several options for implementing a database, such as SQL Server, MySQL, PostgreSQL, and others. In this case, Microsoft SQL Server was used to create a database that will store information related to products and orders.

Entity Framework Core is a powerful Object-Relational Mapping (ORM) tool that can be used to map models to database tables. It provides a Code First approach that allows developers to define database models as classes and then automatically generate the database schema from those classes. This makes it easier to manage database changes and allows developers to work with the database schema in code.

Using Entity Framework Core, the database schema can be created by defining a DbContext that represents the database and by creating classes that define the tables in the database. The DbContext is responsible for communicating with the database and performing database operations such as querying, inserting, updating, and deleting records.

In this case, the Code First approach was used to create a database schema based on the models for products and orders. This involves defining the models as classes with properties that represent the columns in the database tables. Entity Framework Core then uses these classes to generate the database schema.

Once the database schema is created, the ASP.NET Web API solution can use Entity Framework Core to communicate with the database and perform operations such as creating, updating, deleting, and retrieving records. The solution can use the DbContext to query the database and retrieve records as objects, which can then be serialized and returned to the client application as JSON or XML.

Overall, using Microsoft SQL Server and Entity Framework Core with a Code First approach is an effective way to persist data in an ASP.NET Web API solution. It provides a robust and scalable solution that can handle large amounts of data and provides a flexible way to manage database changes.

In order to access the database using Entity framework a number of nuget packages must be downloaded.

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools

**Code First Approach**

In order to create and update a database a number of steps were taken in visual studio.

1. Open Visual Studio and create a new project.
2. In the Solution Explorer, locate the appsettings.json file and open it.
3. Add a new property called “ConnectionStrings” with a value "DefaultConnection" to the JSON object. Set the value of this property to the following string, replacing "IntuitionWebApp" with the name you want to give your database:

*"DefaultConnection": "server=(localdb)\ProjectModels;database=IntuitionWebApp;"*

1. In the Solution Explorer, locate the Program.cs file and open it.
2. Add the following code immediately after builder:

*builder.Services.AddDbContext<ApplicationDbContext>(options => { options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")); });*

1. Save the changes to the appsettings.json and Program.cs files.
2. Open the Package Manager Console by clicking on "Tools" > "NuGet Package Manager" > "Package Manager Console".
3. In the Package Manager Console, run the following command to create an initial migration:

*Add-Migration InitialCreate*

Replace "InitialCreate" with the name you want to give the migration.

1. After the migration is generated, run the following command to apply the migration and create the database:

*Update-Database*

1. You should now have a new database named IntuitionWebApp on your local SQL Server instance.
2. To update the database schema in the future, simply make changes to your entity classes and then generate a new migration using the Add-Migration command. After generating the migration, run the Update-Database command to apply the changes to the database.

**Database first approach**

A Microsoft sql database was exported to a .sql file. In order to create a database using the file the following steps can be taken.

1. Open Microsoft SQL Server Management Studio.
2. Connect to the SQL Server instance where you want to create the database.
3. In the Object Explorer pane, right-click on Databases and select "New Database".
4. In the New Database dialog box, enter a name for the new database and specify the owner of the database (this is typically the database administrator). Click OK to create the new database.
5. Open a new query window by clicking on "New Query" in the toolbar.
6. In the query window, click on "File" in the toolbar and select "Open".
7. Navigate to the location where the exported **Intuitionscript.sql** file is saved and select it.
8. The contents of the .sql file will be displayed in the query window. Make any necessary modifications to the script.
9. Click "Execute" or press F5 to execute the script and create the database.
10. Once the database is created, you can use it with your ASP.NET Web API solution by connecting to the database and performing database operations such as querying, inserting, updating, and deleting records.