The difference between ado.net and entity framework in software development

Miloš ILIĆ[1], Nebojša DENIĆ[2], Dragan ZLATKOVIĆ[3], Jelena STOJANOVIĆ[3], Boban SPASIĆ[1]
[1]Faculty of Information Technology (Alfa BK University, Belgrade, Serbia), email: milos.ilic@alfa.edu.rs and bobanspasic11@gmail.com
[2]Faculty of Sciences and Mathematics (University of Pristina, Kosovska Mitrovica, Serbia), email: nebojsa.denic@pr.ac.rs
[3]Faculty of Mathematics and Computer Science (Alfa BK University, Belgrade, Serbia), email: dragan.zlatkovic@alfa.edu.rs and jelena.stojanovic@alfa.edu.rs

***Abstract*—**This paper shows the difference in data access technologies, ADO.NET and Entity Framework, for the purpose of an actual project, an application for a dentist's office. This application was developed using both technologies in two separate projects, in order to compare the two when working on an actual project, from creating a database, tables, primary and foreign keys, attributes, data types, various constraints, indexes and application development via basic commands for working with a database, such as inserting, reading, updating and deleting data.

***Keywords*—**ADO.NET, Entity Framework, Software development, Visual Studio

I. Introduction

APPLICATION software can be described as end-user software and is used to accomplish a variety of tasks [1]. Modern applications can hardly be imagined without a database [2]. A database is a collection of related data [3]. Access, transport, storage, and manipulation of data are the fundamental and underlying processes at the heart of every information communication technology application/system [4]. In order to allow the application to be linked to the database during development, it is necessary to select one of the data-processing technologies. Two major Microsoft technologies for data access in software development are ADO.NET and Entity Framework.

ADO.NET is a data access technology on the Microsoft .NET platform, which is an important bridge between applications and databases [5]. ADO.NET represents a large set of .NET classes that enable us to retrieve and manipulate data and update data sources, in very many different ways [6]. ADO.NET is Microsoft's core data access library for .NET developers and is the heart of many data-centric technologies on the Windows development platform [7].

J. A. Blakeley, D. Campbell, S. Muralidhar, and A. Nori argue that modern applications and data services need to target a higher-level conceptual model based on entities and relationships rather than the relational model, and that such a conceptual model needs to be implemented concretely in a data platform [8].

Entity Framework is a technology designed to elevate the level of abstraction in data programming from logical (relational schema) to conceptual in which application developers work when creating and maintaining data-centric applications [9]. Entity Framework provides a flexible mechanism for mapping higher-level application models to existing relational schemas [10]. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data and can create and maintain data-oriented applications with less code than in traditional applications [11].

In this paper, a comparison of these two Technologies, ADO.NET and Entity Framework, was made. The methods of adding databases to the project are presented, and the results of the comparison are given for tables, data types, etc.

Creating, reading, updating and deleting (CRUD) operations are principal functions for data-oriented application [12]. ADO.NET and Entity Framework is responsible for these operations. It was precisely on these basic operations that code length was measured, as well as code execution speed.

II. Software Development

The process of software development is called a "software lifecycle", because it represents the description of the software from its inception all the way to its deployment, meaning delivering and maintaining it. Software development can be split into a couple of phases [13]:

1) analyzing and defining the request;
2) system design;
3) software design;
4) software development;
5) software testing;
6) system deployment design;
7) maintenance.

The software development team is in charge of developing software, which comprises analysts, designers, developers, QA engineers and engineers responsible for delivery and maintenance. Depending on the size of the software being developed, it also depends on how many people will be part of the development team [14].

Everybody in the team has their role. The analyst is the person responsible for communicating with the buyer of the software and establishing their demands. Based on the demands established by the analyst, the designer is responsible for software design. Developers are responsible for programming what the designer designed in a specific programming language. The QA engineer then tests this code and thus checks if the software does what it was designed to do. Finally, the delivery and maintenance engineer delivers and installs this software to the end user. Their responsibility is also to educate users on how to use the software and to maintain it after delivery.

Software development involves programming or coding in some of the development tools such as Visual Studio, which was also used in this project.

III. Visual Studio

Visual Studio is an integrated development environment developed by Microsoft that can be used for editing, debugging and building code [15].

It can be used to develop desktop, web and mobile applications, games, web services, business intelligence solutions, etc. The following projects can be created: Console application, Windows Forms application, Web application, Windows API, Windows Presentation Foundation (WPF), Windows Store and Microsoft Silverlight. Visual Studio supports multiple programming languages. The first version of this development tool is Visual Studio 97 which came out in February 1997, while the latest version is Visual Studio 2019, which came out on February 27, 2019.

Visual Studio 2019 is probably the most advanced integrated development environment available to developers today [16].

IV. Application

The application for the dentist's office was developed in Visual Studio 2019 in the Windows Forms Application project in the C# programming language. The purpose of the application is to facilitate work in the dentist's office.

The application's list of features is the following:

1) login for dentists;
2) view timetable;
3) insert, view, update and delete services;
4) insert, view, update and delete dentists;
5) insert, view, update and delete patients;
6) schedule patient appointments;
7) insert and review all prior patient medical procedures.

Special attention should be paid to the database during software development because it represents the basis of each information system [17, 18]. The database is defined as a well-structured collection of data, which is used and maintained by multiple users or applications [19]. It can be said that designing a database is one of the most important parts of information system development [20, 21]. The dentist's office application database is depicted in Figure 1.

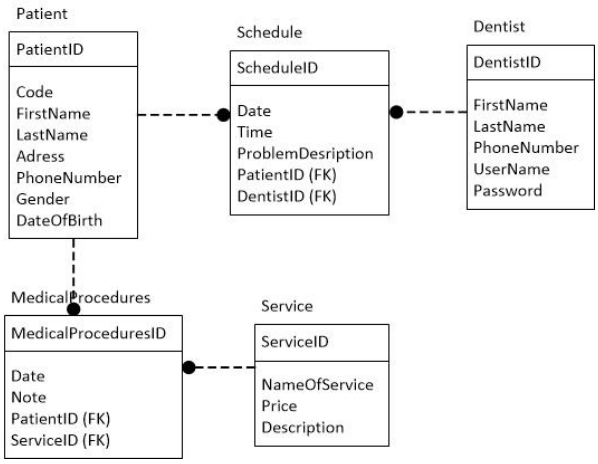

Fig. 1. Database of the dentist's office application

V. ADO.NET

ADO.NET is a .NET data access technology. This technology contains classes that allow us to work with data from the database, to connect to the database, to insert, update and delete data. These classes are called Provider classes [22].

ADO.NET supports two architectures:

1) connected - in which we are constantly connected to the database until the connection is closed. Up-to-date data is a great advantage, while the biggest flaw is that there must be a permanent connection and during that time the rest of the users cannot use it.

2) disconnected - in which we are not permanently connected to the database. Data in this mode is not always up-to-date, but at the same time allows us to let other users use the connection.

In this paper, as far as ADO.NET is concerned, the connected architecture was used for the development of the application.

VI. Entity Framework

Entity Framework represents Microsoft's technology in .NET technology that enables us to access and to process data in relational databases used in application development [23].

Entity Framework is based on the fact that, instead of directly managing tables and data from the database, developers must handle the data in the form of objects and certain properties during software development [24].

There are 4 basic types of approaches to database development:

1) Model First - allows creating a new empty database based on the model we designed in the designer and automatically generating classes based on that model.
2) Database First - implies the use of an existing database, based upon on which classes will be automatically generated.
3) Code First (New Database) - is the approach where you first define classes and mapping by using code to create a model, and then the database is created based on it. It also enables us to change the model during work by using migrations.
4) Code First (Existing Database) - also defines a model based on the code by making classes, but this time the existing database is mapped.

VII. The Methodological Framework Of The Research

In order to see the similarities and differences between the two technologies ADO.NET and Entity Framework, a comparative method was used based on which certain conclusions were drawn. The aim of the research is to determine the advantages and disadvantages of the two technologies for database access and to define in which cases it is better to use ADO.NET, and in which Entity Framework. For this reason, the dentist's office application was developed using ADO.NET in one project, and Entity Framework in the other.

VIII. Comparison of ADO.NET and Entity Framework

A. Databases in the project

When we use the ADO.NET method to access the database, we can add two different databases to the project:
1) Service-based Database
2) DataSet

When we use the Entity Framework method to access the database we can add four different databases to the project:
1) Empty EF Designer model - Model First
2) EF designer from database - Database First
3) Empty Code First model - Code First (New Database)
4) Code First from database - Code First (Existing Database)

Both methods provide more options for adding a database to a project. An already existing database can also be used for the project. If a database does not exist, then there are two options, to start from scratch with a designer or, if it is preferred, to manually code a new one.

B. Data Types

ADO.NET uses SQL data types, while Entity Framework uses C# data types.

The data types for text, numerals, date and time are basically similar and have almost equivalent values, only they have some different tags.

The basic data types used in ADO.NET and its corresponding types in Entity Framework are shown in Table 1.

TABLE I
Data Types

ADO.NET (SQL)	Entity Framework (C#)
char, varchar, nchar, nvarchar	string
bigint	long
int	int
smallint	short
tinyint	byte
bit	bool
decimal	decimal
float	double
real	Single
time	TimeSpan
date, datetime, datetime2, smalldatetime	DateTime

C. Tables

The majority of methods for adding a database to Visual Studio, both in ADO.NET and Entity Framework, rely on graphic design when creating tables or generating them automatically from an existing database. Only in ADO.NET Service-based Database and Entity Framework Code First (New Database), new code is created for tables i.e. classes. It is these two methods, where new code is necessary, that shall be compared.

By comparing those tables that are coded in the ADO.NET Service-based Database code and the Entity Framework Code First (New Database) code, we come to the conclusion that creation is very similar, with differences arising only in syntax, since SQL tables are used for ADO.NET, and C# classes that are equivalent to tables for Entity Framework. Attributes and primary keys are created inside them, for which the same data types (some have different tags) are used, with various settings and constraints. SQL offers the possibility of adding ordinary primary and composite primary keys both inside and outside the tables, while in Entity Framework they must be emphasized in the table itself. In terms of creating keys, some Entity Framework advantages are that, when placing the ID at the end of the attribute name, it is seen as a key and it is not necessary to emphasize it further, and that its automatic generation is implied and that it is obligatory, thus saving time. For ADO.NET i.e. SQL, however, you must emphasize which attribute is the key, whether it is necessary to automatically generate it and whether it is obligatory. When it comes to adding a foreign key, ADO.NET i.e. SQL has an advantage because it is necessary to add only one code for a given table, while in Entity Framework it is necessary to type the code both in the class in which it is located, and in the class from which it originated. When it comes to setting limitations for textual, numerical, and date data types, Entity Framework,

however, allows you to use less code using Data Annotations, in which you can set the desired value directly for the attribute. For indexes that can be clustered and nonclustered (both can be unique and non-unique), a simple nonlustered index is easier to set up using Entity Framework, while the clustered index is far easier to set up in ADO.NET i.e. SQL. When it is necessary to define a column that will be unique but fast searching is not necessary, i.e. no unique index is required, it is something that is supported only by ADO.NET i.e. SQL by setting unique keys, but cannot be set using Entity Framework code. So the only solution for the Entity Framework is setting up a unique index that allows a faster search.

D. Length of Code for Working with the Database

Entity Framework code for working with the database compared to ADO.NET is shorter and simpler, which is a great advantage. When it comes to working with basic commands over the database in the project, such as inserting, reading, updating and deleting data, by comparing these two technologies, there is a big difference in code length, where in some cases it is twice as short. The comparison of code length for basic database operations for ADO.NET and Entity Framework in the part of the application dealing with patients can be seen in Table 2. The results are expressed in the number of lines of code needed for coding.

TABLE II
LENGTH OF CODE ADO.NET VS ENTITY FRAMEWORK

Operation	ADO.NET	Entity Framework
Insert	20	18
Read	12	5
Update	21	17
Delete	7	4

In addition to the fact that the Entity Framework has fewer lines of code than ADO.NET, it should be noted that every single line of its code is shorter to write as well.

Entity Framework makes working with the database more comfortable than ADO.NET does. It is not necessary to write long queries and the such. Much happens "under the hood", which would otherwise need to be coded in ADO.NET. Entity Framework significantly reduces the redundancy of the code, which leads to its much easier maintenance.

Since it requires less implementation code, Entity Framework means much faster software development, which is of great importance today, as it is always required that applications be completed as soon as possible.

E. Code execution speed

Speed is one of the most important features of the application, so we should strive to make its performance as quick as possible. Of course, this also depends on the technology we use in its development. In order to compare code execution speed between ADO.NET and Entity Framework, part of the application for inserting, modifying and deleting a patient was taken as an example and code that will allow timing was created.

After launching the application, 10 patients were inserted, one at a time, first for ADO.NET technology, and then 10 of the same patients again for Entity Framework. After each entry, time was recorded. Then, the 10 patients were updated and deleted and the results were also recorded.

The results of executing the code for insert, update and delete, converted in seconds for ADO.NET and Entity Framework are shown in Table 3.

TABLE III
EXECUTION SPEED ADO.NET VS ENTITY FRAMEWORK

No.	ADO.NET			Entity Framework		
	Insert	Update	Delete	Insert	Update	Delete
1	0.023	0.005	0.024	0.115	0.034	0.160
2	0.002	0.003	0.001	0.013	0.007	0.004
3	0.010	0.003	0.001	0.013	0.005	0.004
4	0.002	0.003	0.001	0.025	0.007	0.004
5	0.002	0.003	0.001	0.013	0.005	0.004
6	0.002	0.003	0.001	0.013	0.005	0.004
7	0.008	0.003	0.001	0.018	0.005	0.004
8	0.002	0.003	0.001	0.013	0.005	0.004
9	0.002	0.003	0.001	0.013	0.005	0.004
10	0.002	0.001	0.001	0.013	0.005	0.004

As can it be seen on the displayed results, ADO.NET technology is definitely faster, that is, its performance is better than Entity Framework.

In both cases, it takes a lot more time for the initial execution of the code, but every subsequent execution is faster.

The average time to insert 10 patients for ADO.NET is 0.0055 seconds, while the time for Entity Framework is greater, at 0.0249 seconds, which means that, according to these results, ADO.NET is faster by as much as 4.53 times compared to Entity Framework.

When it comes to updating 10 patients, the average time for ADO.NET is 0.003 seconds, and for Entity Framework it is 0.00524. ADO.NET is also 1.75 times faster here.

Deleting 10 patients has the following results, ADO.NET average time is 0.0033, and Entity Framework 0.0196, which again means that ADO.NET is 5.94 times faster.

This proved that ADO.NET is faster than Entity Framework. The reason lies in the fact that Entity Framework technology is built upon ADO.NET and therefore cannot be faster.

F. Migrating applications to another service provider

ADO.NET does not have its own query language but uses SQL queries that directly forward to the service provider, without their prior manipulation. Entity Framework has its own queries written in LINQ or Entity SQL, which are then translated into some background syntax for that database by the service provider. So if there were a need to move to another provider, such as Oracle,

and ADO.NET was used to develop software, a large number of queries would have to be changed, while Entity Framework queries would not need to be changed.

IX. Conclusion

Both technologies provide multiple ways to add a database to a project, and are similar in some regards. The types of data used to define attributes are also similar. When creating a table, the syntax is quite different, because SQL tables are used for ADO.NET and C# classes for Entity Framework. Primary and foreign keys, constraints and indexes have the same functions but are added in completely different ways. Working with the database, on the examples of inserting, displaying, updating and deleting, has shown that there is a big difference in code length. Entity Framework code is much shorter, simpler, and provides greater flexibility in working with the code than ADO.NET code, but it also does not allow for better performance of the application, since Entity Framework itself is built over ADO.NET and cannot be faster than it. Switching from one provider to another, e.g. from SQL Server to Oracle is much easier when the application is developed with Entity Framework technology because then there is no need to change a huge number of queries, which would otherwise have to be changed if ADO.NET were used. Finally, if our priority is application performance, the right choice for development would be ADO.NET. On the other hand, if a quicker development cycle and an easier transition from one provider to another are the priority, then Entity Framework should be used.

References

[1] T. J. O'Leary, and L. I. O'Leary, *Computing Essentials: Making IT work for you, Complete 2014 Edition*. The McGraw-Hill, New York, 2014, pp. 64

[2] N. Denic, V. Vujovic, V. Stevanovic, and B. Spasic, *Key factors for successful implementation of ERP systems*. The journal Tehnički Vjesnik/Technical Gazette, vol. 23, no. 5, pp. 1335-1341, October 2016. ISSN 1330-3651, DOI: 10.17559/TV-20150618213311, IF 0,723 (2016), M23.

[3] M. Ilić, L. Kopanja, D. Zlatković, M. Trajković, and D. Ćurguz, *Microsoft SQL Server and Oracle: Comparative performance analysis*. In: Book of proceedings of the 7th International conference Knowledge management and informatics, Vrnjačka Banja, pp. 33-40, june 2021.

[4] T. Vaikunth Pai, and P. S. Aithal, *Disconnected Data Access Architecture using ADO.NET Framework*. International Journal of Applied Engineering and Management Letters (IJAEML), vol. 1, no. 2, pp. 10-16, 2017.

[5] J. Wang, *Web Database Access Technology Based on ASP.NET*. In: D. Jin, S. Lin (eds) Advances in Multimedia, Software Engineering and Computing vol.1. Advances in Intelligent and Soft Computing, vol 128. Springer, Berlin, Heidelberg, 2011.

[6] J. Skinner, B. Joshi, D. Mack, D. Seven, F. C. Ferracchiati, J. Narkiewicz, J. McTainsh, K. Hoffman, M. Milner, and P. Dickenson, *Professional ADO.NET Programming*, Apress, 1st edition, p. 1, 2001.

[7] T. Patrick, *Microsoft ADO.NET 4 Step by Step*. O'Reilly Media, Inc., Sebastopol, CA, pp. 17, 2010.

[8] J. A. Blakeley, D. Campbell, S. Muralidhar, and A. Nori, *The ADO.NET Entity Framework: Making the Conceptual Level Real*. SIGMOD Record, vol. 35, no. 4, pp. 31–38, 2006.

[9] A. Adya, J. Blakeley, S. Melnik, and S. Muralidhar, *Anatomy of the ADO.NET Entity Framework*. Proceedings of the 2007 ACM SIGMOD International Conference of Management of Data, vol. 2, pp. 877-888, 2007.

[10] P. Castro, S. Melnik, and A. Adya, *ADO.NET Entity Framework: Raising the Level of Abstraction in Data Programming*. Proceedings of the 2007 ACM SIGMOD International Conference of Management of Data, vol. 2, pp. 1070-1072, 2007.

[11] M. Wadhwa, *A comparative study between ado.net and entity framework*. International Journal of Advanced Scientific and Technical Research. vol. 6, no. 7, pp. 154-159, 2017.

[12] X. Yang, and H. Shen, *A Unified Design and Implementation for Creating, Reading, Updating and Deleting Operations Based on ADO.NET Entity Framework*. In: M. Zhao, J. Sha (eds) Communications and Information Processing. Communications in Computer and Information Science, vol 289. Springer, Berlin, Heidelberg, 2012.

[13] V. Tomašević, *Razvoj aplikativnog softvera*. Univerzitet Singidunum, Beograd, p. 16, 2012.

[14] N. Denic, B. Dasic, and J. Maslovara, *Profitability of the investment project of introducing modern business information systems*. TTEM - Technics Technologies Education Management, vol. 8, no. 1, pp. 367-372, 2013. ISSN: 1840-1503, e-ISSN: 1986-809X. IF 0.414 (ISI Journal Citation Reports 2012).

[15] G. Warren, and T. G. Lee, *Welcome to the Visual Studio IDE*. https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2017, accessed on March 2021.

[16] D. Strauss, *Getting Started with Visual Studio 2019: Learning and Implementing New Features*, Apress, pp. 25, 2019.

[17] N. Denic, V. Vujovic, A. Skulic, and S. Filic, *Studious analysis of business intelligence systems in Serbian enterprises*. 16th International Multidisciplinary Scientific GeoConference & EXPO SGEM 2016, vol. 1, pp. 425-436, 2016. ID: 156242016032699992, ISSN: 1314-2704.

[18] N. Denić, V. Vujović, S. Filić, and B. Spasić, *Analysis of key success factors for business intelligence systems implementation*. Annals of the University of Oradea, Fascicle of Management and Technological Engineering, no. 1, 2016. ISSN 1583 - 0691, CNCSIS "Clasa B+".

[19] B. Lazarević, Z. Marjanović, N. Aničić, and S. Babarogić, *Baze podataka*, Fakultet organizacionih nauka, Beograd, peto izdanje, p. 1, 2010.

[20] N. Denić, D. Petković, and B. Spasić, *Global Economy Increasing by Enterprise Resource Planning*. Editor(s): S. Hashmi, I. A. Choudhury, Encyclopedia of Renewable and Sustainable Materials, Elsevier, vol. 1, pp. 331-337, 2020. ISBN 9780128131961, https://doi.org/10.1016/B978-0-12-803581-8.11590-5.

[21] N. Denić, S. Marković, B. Spasić, and M. Milić, *Identification of influential factors of project implementation information systems*. Annals of the University of Oradea, Fascicle of Management and Technological Engineering, vol. 23, no. 13, pp. 123-126, 2014. ISSN 1583 - 0691, CNCSIS "Clasa B+", DOI: 10.15660/AUOFMTE.2014-2.3090.

[22] G. Aritonović, *Objektno programiranje*, Beogradska poslovna škola, Beograd, p. 168, 2010.

[23] B. Driscoll, N. Gupta, R. Vettor, Z. Hirani, and L. Tenny, *Entity Framework 6 Recipes*, Apress, New York, NY, 2nd Edition, pp. 2, 2013.

[24] B. Kolar, and V. Blagojević, *ENTITY FRAMEWORK za data-orijentisane aplikacije*. https://www.helloworld.rs/blog/ENTITY-FRAMEWORK-za-data-orijentisane-aplikacije/289, accessed on July 2021.