UDC: 004.72:004.774
Original scientific paper

A COMPREHENSIVE ANALYSIS OF ARCHITECTURAL PATTERNS IN ASP.NET CORE WEB APPLICATION DEVELOPMENT

Florinë MUSHICA[1], Agon MEMETI[1*]

[(1)] University of Tetova, Faculty of Natural Sciences and Mathematics, Tetovo; [(2)] UBT College, Prishtina North Macedonia
** Corresponding Author, e-mail: agon.memeti@unite.edu.mk*

Abstract

The landscape of web application development has undergone significant transformations since the inception of the World Wide Web. In the contemporary landscape of web development, a multitude of design patterns, frameworks, and programming languages are employed to construct web applications, adapting to their diverse levels of complexity. Model-View-Controller (MVC) stands out as a widely embraced design pattern in the construction of web applications. This paper's primary goal is to provide a comprehensive elucidation of the features and fundamental components intrinsic to the MVC architectural pattern. Through practical examples and in-depth analysis, readers will gain valuable insights into the nuances of MVC, empowering them to harness its capabilities for robust and effective web application development.

Keywords: MVC, model, view, ASP.NET, controller, architecture.

1. Introduction

Web applications today have become essential components that cover many fields such as education, art, entertainment, business, and many others. With the rapid progress of this paradigm, the ways in which we use and interact with web applications have changed a lot. In such a scenario, the development of web applications has resulted in a continuous expansion of the complex and dynamic demands of users. Therefore, the efficiency of web application development is very critical. Web developers today face an even greater challenge in choosing the right technology to accommodate all user requests.
A good practice during development is to divide the application's structure into components or layers and make the development modular. This combination results in a scaled and very flexible architecture for web applications. Several design models offer this type of solution as templates for best practices for the development of web applications and real-world systems. Nowadays, the focus is on the MVC model, which has become very popular for designing applications due to its structural characteristics and adaptation to programming languages. Its main characteristic is its logical model, view, and controller components, which deal with specific aspects of web application development.
The objective of this study is to cover the architecture of web applications, more precisely the MVC architectural pattern, and explain its functionalities on a practical basis through the development of an ASP.NET core web application. The application development process is supported by the following systems, technologies, and frameworks: Visual Studio (IDE), ASP.NET core MVC, Cionalities on a practical basis through the development of an ASP.NET core web application. The application development process is supported by the following systems, technologies, and frameworks: Visual Studio (IDE), ASP.NET core MVC, C#, and several Microsoft packages such as EntityFrameworkCore, SQLServer, Tools, and AspNetCore. Identity.

Furthermore, this paper is organized as follows:

- Chapter 2 State of the Art, which basically has a review of literature and published articles on the topic,
- Chapter 3 provides an overview of Web application design and development,
- Chapter 4 offers website UI design, the latter followed by conclusions.

2. State of the Art

According to several authors [1], design models are very important elements in the development of web and mobile applications. Over the years, web applications have evolved along with their complexity; for this reason, these problems must be addressed through the proper selection of architectural models and frameworks.
MVC (Model-View-Controller) architecture, as a result of the facilitating structure that it offers developers, is making it widely used for the development of web applications on the Internet. This strict decomposition of applications into three components: model, view, and controller are making the applications more scalable, efficient, and maintainable.
According to Patel and Pancholi [2], this division of this architectural model is made to separate the way of presentation and acceptance of information by the user from its internal representations. It also enables parallel development and efficient code reuse. Background information: The MVC architecture model was presented by the computer scientist Trygve Mikkjel Heyerdahl Reenskaug for the first time in 1979 [3]. His goal was to come up with a model that divided a complex application into smaller and more manageable components. This model was first used in the small talk programming language. The original designation for this model was Model-View-Editor, but later it was designated Model-View-Controller. In its beginnings, specifically in the 1980s and 1990s, the MVC model was mainly used for the development of desktop applications. But from the end of the 1990s until today, the MVC model has been a very popular choice within web applications. Each component of the architecture is intended to address specific aspects of web application development. In the following section, we have the description of the components according to several authors [4] of the MVC architectural pattern: model, view, and controller.

The model component is the part of the application that represents all the logic of the data used in the application with which the user works. It corresponds to the data that is transferred between the view and controller components, as well as any other data related to the business logic. Since the controller component cannot interact with the database, the model component responds to every request of the controller. Every operation within the application, such as adding and retrieving data, that requires the presence of the database is managed by the model.
The view component is responsible for generating the user interface of the application. They are created from the data that the model component possesses. This data cannot be obtained directly from the model but rather through the controller component. So, the view component interacts only with the controller.
The controller component is responsible for handling events and acts as an intermediary between the view and model components. Responsible for processing logic and input requests, manipulates data through the model component, and then interacts with the view component to give the result of the right view that will be displayed to the user according to his requests.

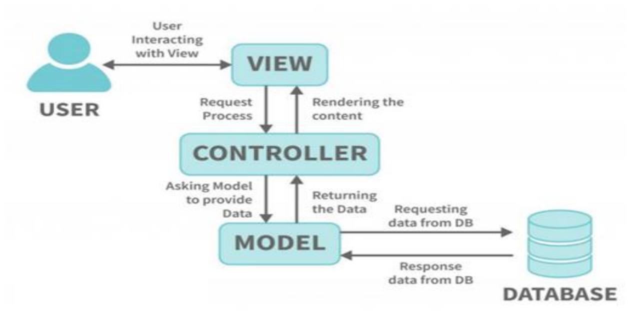


Figure 1. MVC architecture: Data flow [5]

MVC is used in several popular web frameworks, including ASP.NET, Angular, Laravel, Ruby, etc. ASP.NET Core is a high-performance, cross-platform, open-source framework for building cloud-optimized, modern web applications. ASP.NET Core apps are modular and lightweight with built-in support for dependency injection, enabling greater maintainability and testability. Combined with MVC, ASP.NET Core is a powerful framework that supports building modern web APIs in addition to view-based apps [6].

So, as a conclusion to this chapter, when should we apply the MVC pattern architecture? When we are dealing with applications that have a very serious and complex interaction on the part of the client, it is not enough just to use JavaScript. As for the cases where the complexity leans convincingly on the server side and there is very little communication on the client side, then the use of other development models comes into play. Some characteristics that help us understand when we should use the MVC architecture for developing applications are listed below [7]:

- When the same type of data is distributed in different ways on a single page.
- Data manipulation is generally performed by the client and not by the server.
- When the application has functionalities that must be realized without reloading the page completely.
- Applications need asynchronous communication on the backend side etc.

3. Web Application Design and Development

In this research study, a web application has been developed using the MVC architecture, through which the basic concepts and logic of this model will be practically clarified. The web application "Foreign Languages Academy" refers to the sale and purchase of the product (a language course) online. The application can be considered a B2C app since the academy that offers courses can be considered a business, while a customer is considered a user (student) who selects one or more products (courses) that are offered.

The web application has implemented the following actions, which are performed by different actors depending on the role:

1. User login or registration
2. The user can search for the product (language course) in which he is interested.
3. The user can see the details of the course (language, level, professor, assistant, description, class, price, start date, end date, and status: coming up, expired, available).
4. The user can add one or more products to the basket and purchase them later.
5. The user can view his shopping cart, add or delete courses, and see the total price before the checkout process.
6. The user with the Admin role also logs into the system and can view orders (transactions) as well as users of the entire web application.

7. The administrator manages the system; he can add, edit, and delete languages, professors, assistants, and classes.

So, the Web application is divided into two functionalities: user and admin, which vary in some operations that are visible depending on the role at the moment of identification. The second component, the admin, mainly includes the application of CRUD (create, read, update, and delete) operations on the application data.

Development: The application was developed using the ASP.NET Core MVC template. Views have been developed using the Razor Engine, the controllers are developed using C# classes (*.cs), and models use entities that are also C# classes.

Features: ASP.NET Core MVC offers many features that are useful for building web-based APIs and apps. Some of the features of this architecture that have been applied to the project are worth noting, such as model binding, dependency injection, routing, authorization, etc.

Model binding in ASP.NET MVC transforms client request data into objects that the controller can handle. As a result, the data is passed as parameters to its action methods, so the controller logic does not have to do the job of determining the incoming request data.

Model validation in ASP.NET Core is done by decorating your model objects with data annotation validation attributes. The validation attributes are checked on the client side before values are sent to the server.

Dependency Injection: in ASP.NET Core, controllers can request required services through their constructors.

Application models overview

A *model* is a c sharp class that represents a table in the SQL server and defines the data relations. Initial models of the application are: professor, assistant, language and class, which represent database tables.

Between the language and professor model, we come to the use of join tables (models) because of the relationship between them (many to many). So, we split the many-to-many relationship into two one-to-many relationships by adding the Professors_Languages join table.

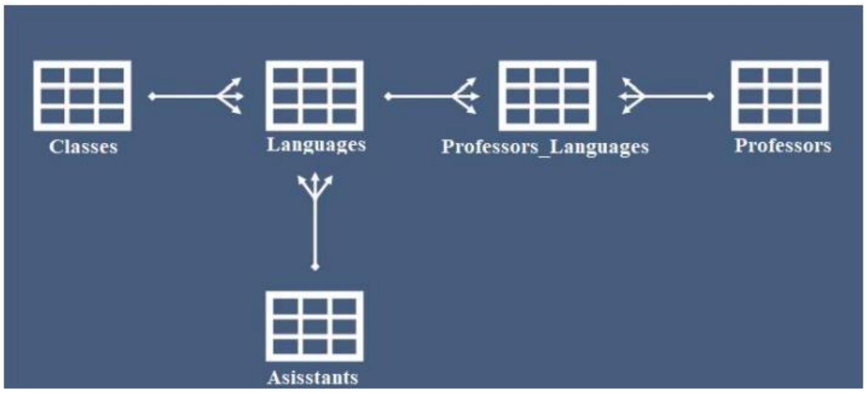


Figure 2. Database schema with initial models

After we have added initial models to the web app, we have to add a DB context file, which serves as a translator between the application model and the database. So for the SQL database to understand the C# code and vice versa, we must have a file in between that understands both C# and SQL.

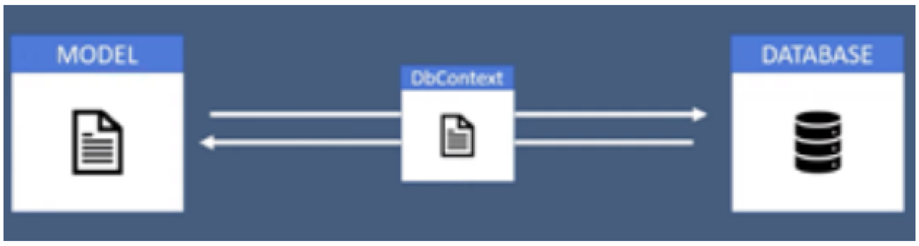


Figure 3. DBContext file

For the many-to-many relationship, we must add some instructions to the dbContext file and define the names of the tables.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
modelBuilder.Entity().HasKey(pl => new
{
pl.ProfessorId,
pl.LanguageId
});
modelBuilder.Entity().HasOne(l => l.Language).WithMany(pl => pl.Professors_Languages).HasForeignKey(l => l.LanguageId);
modelBuilder.Entity().HasOne(l => l.Professor).WithMany(pl => pl.Professors_Languages).HasForeignKey(l => l.ProfessorId);
base.OnModelCreating(modelBuilder);
}
public DbSet Professors { get; set; }
public DbSet Languages { get; set; }
public DbSet Professors_Languages { get; set; }
public DbSet Classes { get; set; }
public DbSet Assistants { get; set; }
//Orders related tables
public DbSet Orders { get; set; }
public DbSet OrderItems { get; set; }
public DbSet ShoppingCartItems { get; set; }
}
```

In addition to the initial models, the web application also contains other models, such as: Order, OrderItem, ShoppingCartItem, and ApplicationUser, which play an important role in the proper functioning of the application.

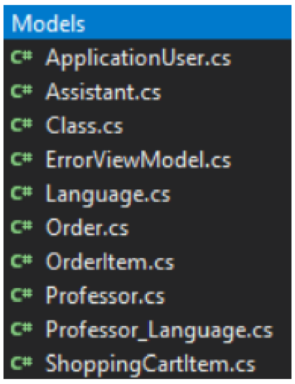
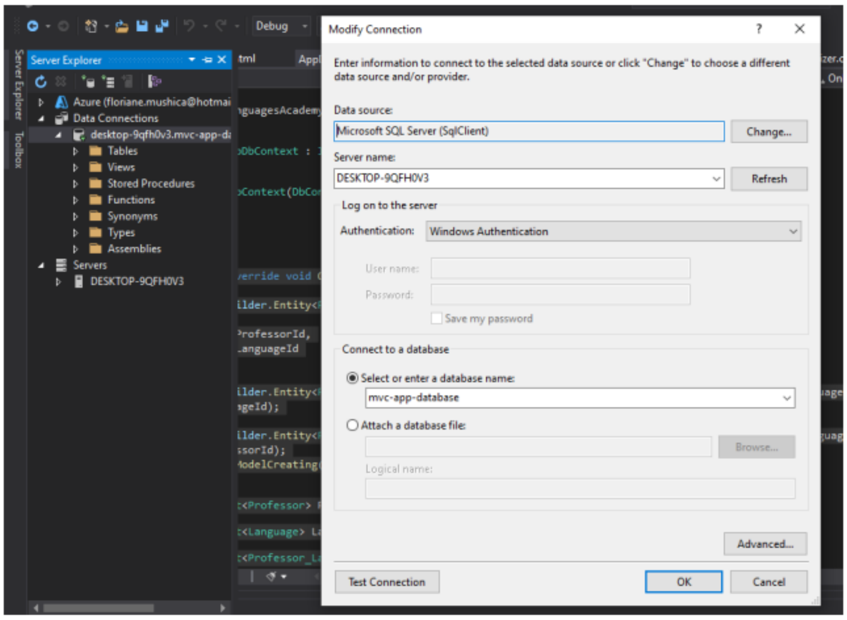

Models
- ApplicationUser.cs
- Assistant.cs
- Class.cs
- ErrorViewModel.cs
- Language.cs
- Order.cs
- OrderItem.cs
- Professor.cs
- Professor_Language.cs
- ShoppingCartItem.cs

Figure 4. Application models

SQL Server configuration

After we have defined the relationships between the models and completed the AppDbCongtext file, the SQL server is configured to be the default storage of the application data.


Modify Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DESKTOP-9QFH0V3

Log on to the server

Authentication: Windows Authentication

Connect to a database

(●) Select or enter a database name: mvc-app-database

( ) Attach a database file:

Figure 5. Database configuration

//DbContext configuration

services.AddDbContext(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnectionString")));

Definition of data storage in startup.cs.

Database: synchronizing model changes with the database In EntityFramework Core, the synchronization of model changes with the database is done by migration. Combining the AppDbContext file with the templates generates a C# class that reflects these changes. After adding the necessary migrations and updating the database, the latter is populated with the necessary data for the operation of the application. After these steps, the database is ready to be seeded with data.

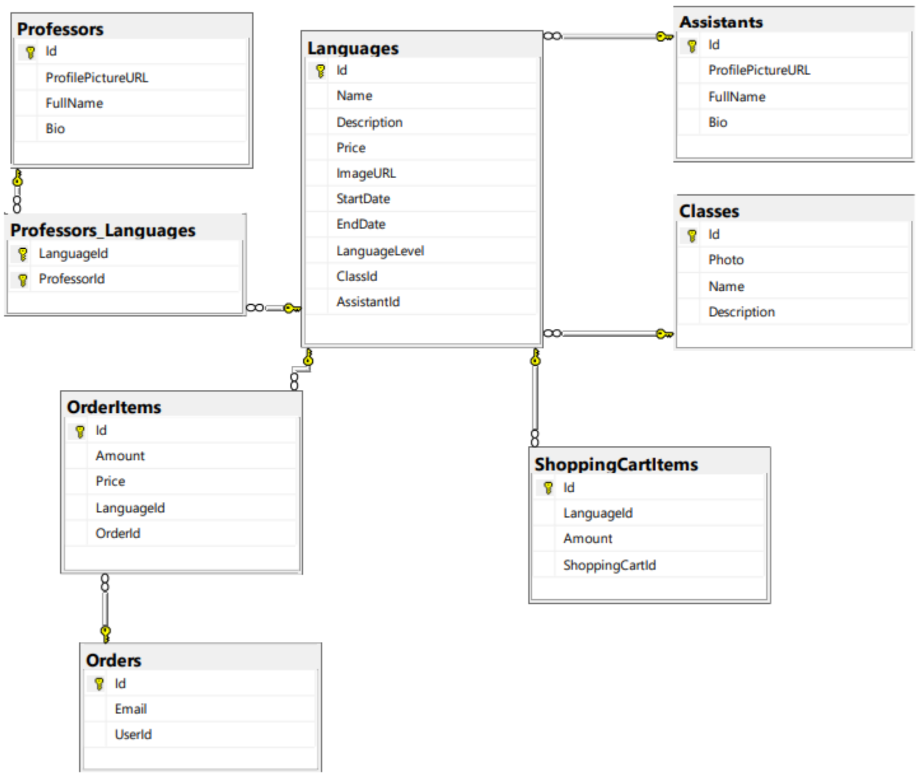


Figure 6. Database diagram

Application controllers overview

A controller in ASP.Net Core is a C# class that is responsible for controlling how a user interacts with an MVC application and handles its requests. The created application contains a controller for each feature. Each controller contains methods that are used for different functionalities within the application's features. Some features can only be seen by the administrator. The MVC controllers of the web application are shown in the figure below.

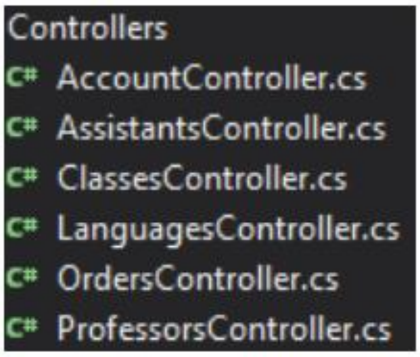
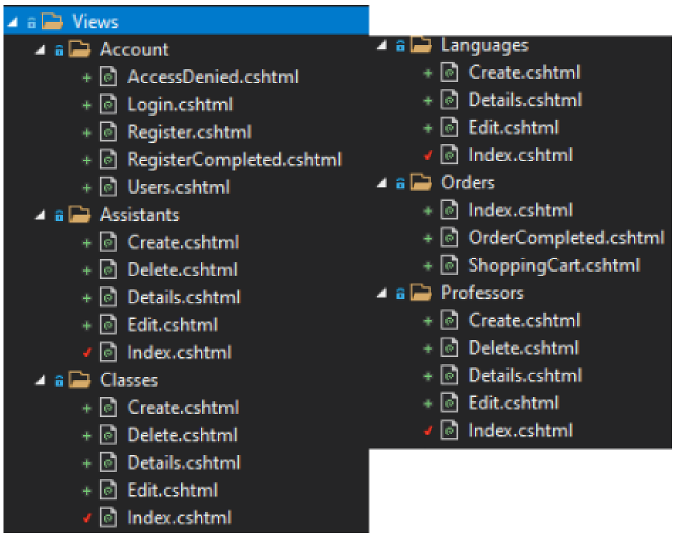

Controllers
AccountController.cs
AssistantsController.cs
ClassesController.cs
LanguagesController.cs
OrdersController.cs
ProfessorsController.cs

Figure 7. Application controllers

Application views overview

The view presents the user interface, in our case the admin user interface and the user (student) interface. In the view, in addition to the html code, we also have the definition of the type of data used in that particular view and c# code. Since in the application certain features can only be seen by the admin, the application is coded in such a way that some parts of the UI change when an if statement is fulfilled.

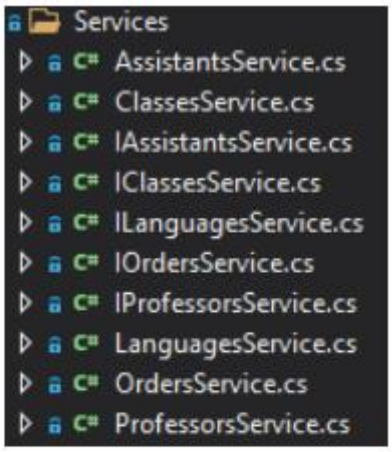
@if (User.Identity.IsAuthenticated && User.IsInRole("Admin")){ code required for certain features here… }



```
Views
  Account
    AccessDenied.cshtml
    Login.cshtml
    Register.cshtml
    RegisterCompleted.cshtml
    Users.cshtml
  Assistants
    Create.cshtml
    Delete.cshtml
    Details.cshtml
    Edit.cshtml
    Index.cshtml
  Classes
    Create.cshtml
    Delete.cshtml
    Details.cshtml
    Edit.cshtml
    Index.cshtml
  Languages
    Create.cshtml
    Details.cshtml
    Edit.cshtml
    Index.cshtml
  Orders
    Index.cshtml
    OrderCompleted.cshtml
    ShoppingCart.cshtml
  Professors
    Create.cshtml
    Delete.cshtml
    Details.cshtml
    Edit.cshtml
    Index.cshtml
```

Figure 8. Application views

A good practice when developing applications is to use services so that the code to receive the data is not written in the controllers. So, in the application, we used services that interact with the database using EntityFramework Core, which are injected into the controller.



```
Services
  AssistantsService.cs
  ClassesService.cs
  IAssistantsService.cs
  IClassesService.cs
  ILanguagesService.cs
  IOrdersService.cs
  IProfessorsService.cs
  LanguagesService.cs
  OrdersService.cs
  ProfessorsService.cs
```

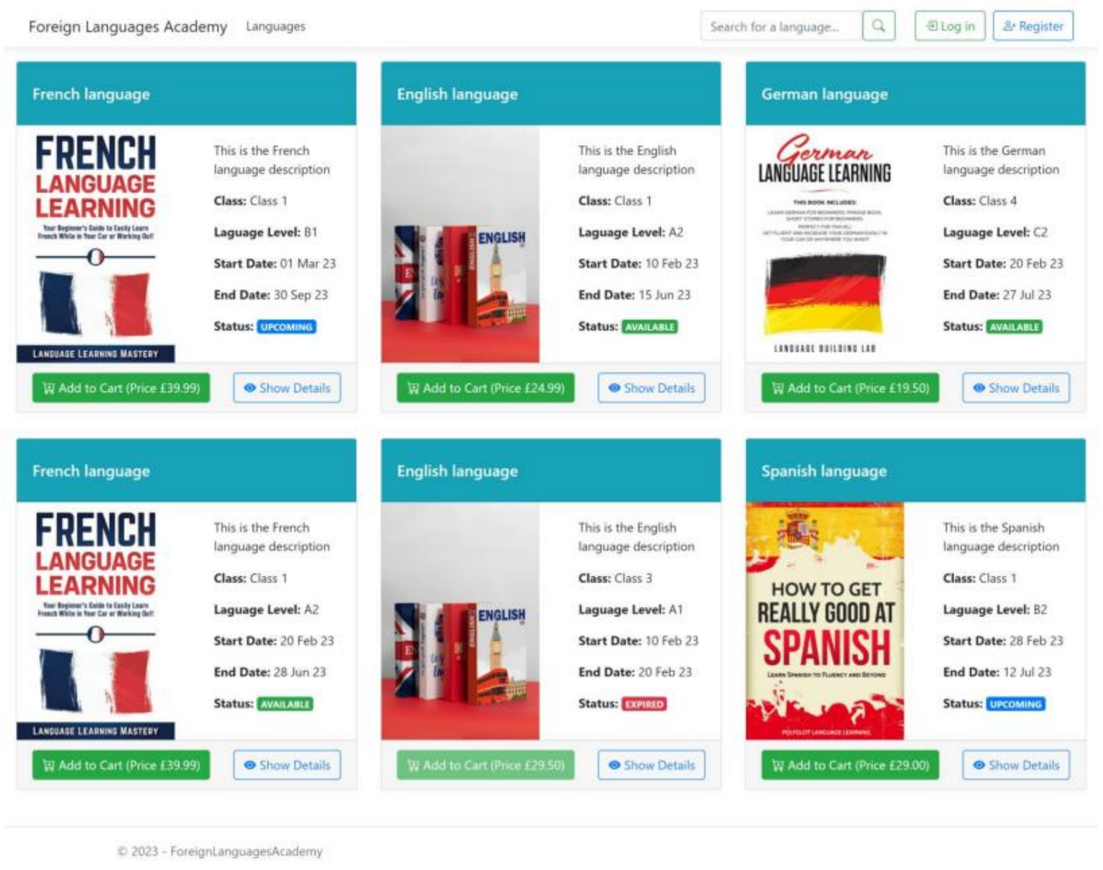
Figure 9. Application services

In cases where the same methods are used for several functionalities (add, update, delete, read), generic solutions are used. In this web application, the administrator performs several operations that use the same methods, such as adding, updating, deleting, getting professors, assistants, languages, etc., which are implemented in a separate file that is inherited by certain services and has as a parameter the necessary model.

4. Web UI Design

As mentioned before, the UI changes in certain views depending on the role of the user, which can be user or admin.

The initial view looks like the following, if we click the add to cart button, it redirects us to the login page.


Foreign Languages Academy Languages Search for a language... Log in Register

French language

This is the French language description

Class: Class 1

Laguage Level: B1

Start Date: 01 Mar 23

End Date: 30 Sep 23

Status: UPCOMING

Add to Cart (Price £39.99) Show Details

English language

This is the English language description

Class: Class 1

Laguage Level: A2

Start Date: 10 Feb 23

End Date: 15 Jun 23

Status: AVAILABLE

Add to Cart (Price £24.99) Show Details

German language

This is the German language description

Class: Class 4

Laguage Level: C2

Start Date: 20 Feb 23

End Date: 27 Jul 23

Status: AVAILABLE

Add to Cart (Price £19.50) Show Details

French language

This is the French language description

Class: Class 1

Laguage Level: A2

Start Date: 20 Feb 23

End Date: 28 Jun 23

Status: AVAILABLE

Add to Cart (Price £39.99) Show Details

English language

This is the English language description

Class: Class 3

Laguage Level: A1

Start Date: 10 Feb 23

End Date: 20 Feb 23

Status: EXPIRED

Add to Cart (Price £29.50) Show Details

Spanish language

This is the Spanish language description

Class: Class 1

Laguage Level: B2

Start Date: 28 Feb 23

End Date: 12 Jul 23

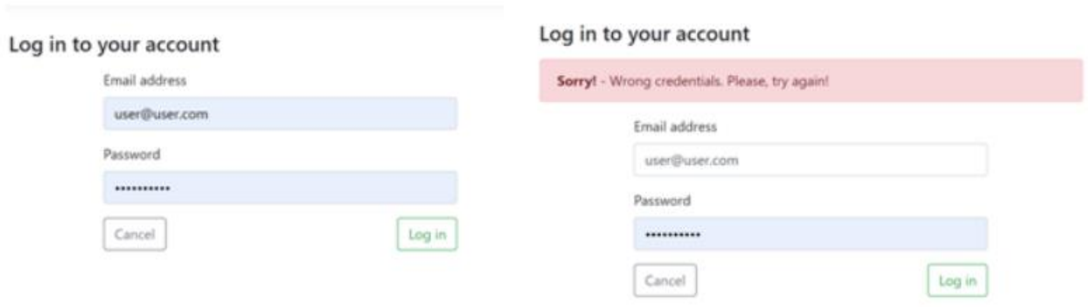
Status: UPCOMING

Add to Cart (Price £29.00) Show Details

© 2023 - ForeignLanguagesAcademy

Figure 10. Website UI design

In the following figure, we see the login page and the case when we give the wrong data.


Log in to your account

Email address: user@user.com

Password: ••••••••••

Cancel Log in

Log in to your account

Sorry! - Wrong credentials. Please, try again!

Email address: user@user.com

Password: •••••••••••

Cancel Log in

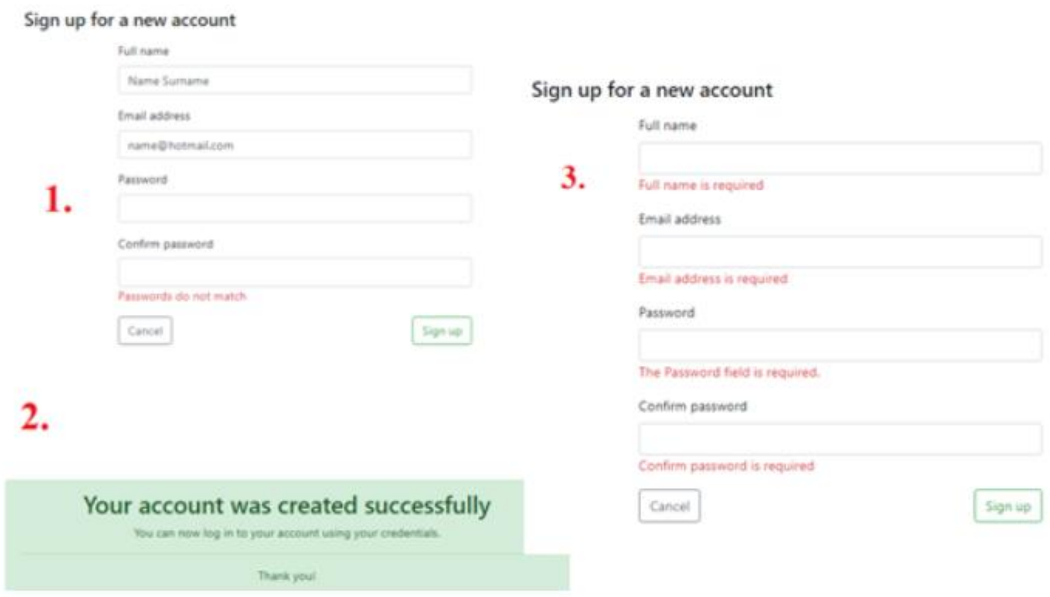
Figure 11. Website UI design-Login

Register page:

1). the case when the password does not match,

2). the case when the account is created successfully,

3) error message when input data is missing.



Sign up for a new account

Full name: Name Surname

Email address: name@hotmail.com

Password

Confirm password

Passwords do not match

Cancel | Sign up

Your account was created successfully

You can now log in to your account using your credentials.

Thank you!

Sign up for a new account

Full name

Full name is required

Email address

Email address is required

Password

The Password field is required.

Confirm password

Confirm password is required

Cancel | Sign up

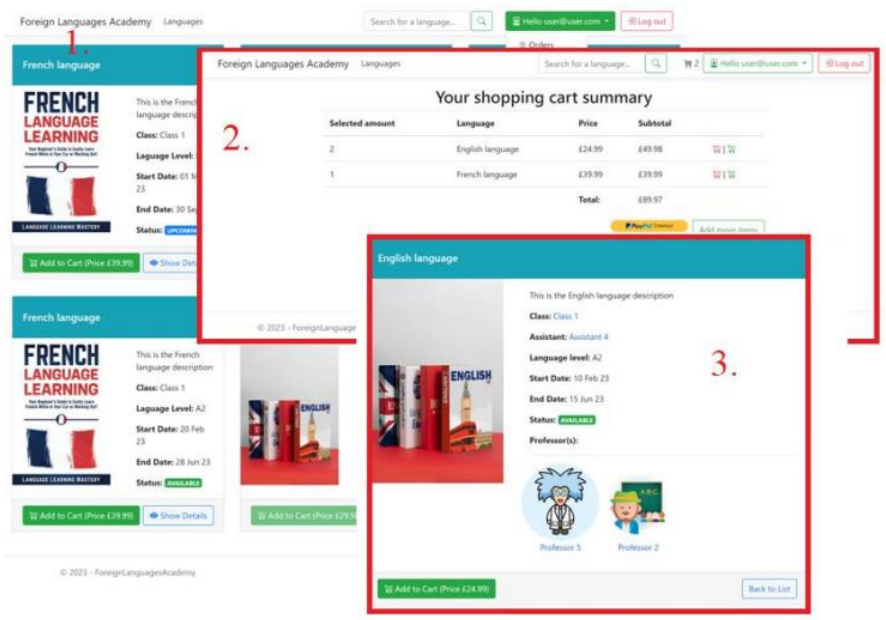
Figure 12. Website UI design-Register

Website UI when the role is identified as USER.

1). the initial view after logging in,

2). the view of the shopping cart after choosing the courses,

3). the view after clicking the show details button



French language

This is the French language description

Class: Class 1

Laguage Level: A2

Start Date: 20 Feb 23

End Date: 28 Jun 23

Status: AVAILABLE

Your shopping cart summary

Selected amount	Language	Price	Subtotal
2	English language	£24.99	£49.98
1	French language	£39.99	£39.99
		Total:	£89.97

English language

This is the English language description

Class: Class 1

Assistant: Assistant 4

Language level: A2

Start Date: 10 Feb 23

End Date: 15 Jun 23

Status: AVAILABLE

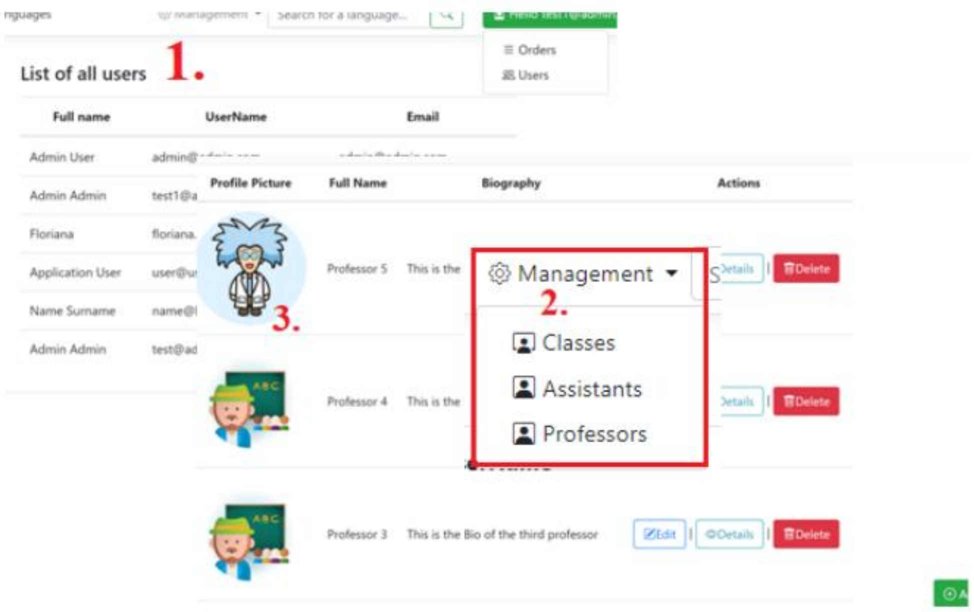
Professor(s):

Add to Cart (Price £24.99)

© 2023 - ForeignLanguagesAcademy

Figure 13. Website UI design-User Role

Website UI when the role is identified as ADMIN. 1). The view after pressing account/users, 2). Dropdown items that can be managed by the admin, 3). the professors view.



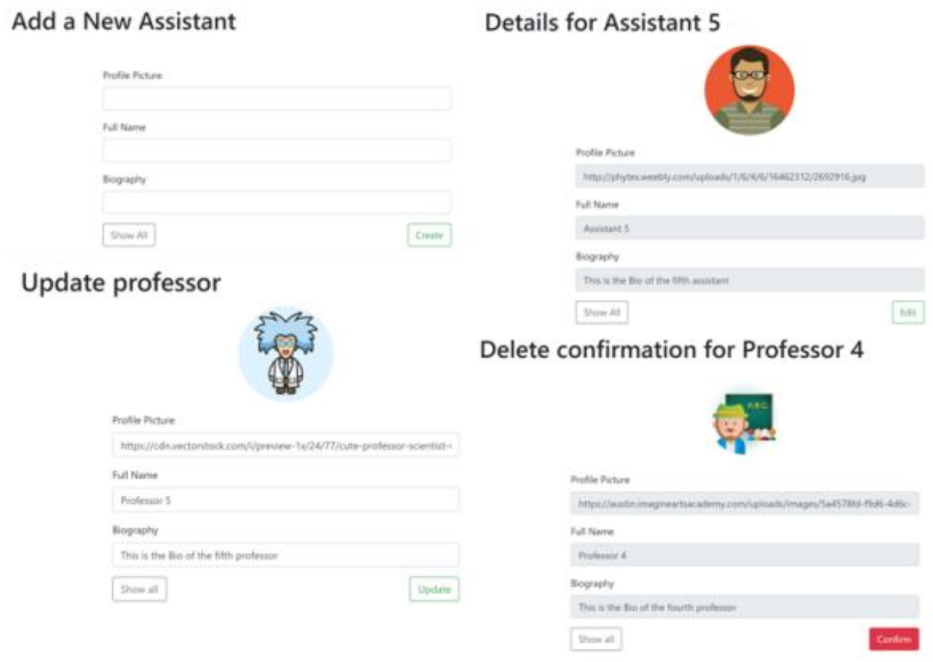
List of all users 1.

Full name	UserName	Email
Admin User	admin@	
Admin Admin	test1@a	
Floriana	floriana.	
Application User	user@u	
Name Surname	name@	
Admin Admin	test@ad	

Profile Picture	Full Name	Biography	Actions
3.	Professor 5	This is the	Details | Delete
	Professor 4	This is the	Details | Delete
	Professor 3	This is the Bio of the third professor	Edit | Details | Delete

Figure 14. Website UI design - Admin Role

In the following figure, some examples of performing CRUD operations within the web application are documented. These operations can be performed on professors, assistants, classes, and languages by the admin.



Add a New Assistant

Profile Picture

Full Name

Biography

Details for Assistant 5

Profile Picture: http://phytes.weebly.com/uploads/1/6/4/6/16462312/2692916.jpg

Full Name: Assistant 5

Biography: This is the Bio of the fifth assistant

Update professor

Profile Picture: https://cdn.vectorstock.com/i/preview-1x/24/77/cute-professor-scientist-

Full Name: Professor 5

Biography: This is the Bio of the fifth professor

Delete confirmation for Professor 4

Profile Picture: https://austin.imagineartsacademy.com/uploads/images/5a4578fd-f9d6-4d6c-

Full Name: Professor 4

Biography: This is the Bio of the fourth professor

Figure 15. Website UI design-CRUD operations

5. Conclusions

This paper presents an overview of the MVC architectural pattern through the development of an ASP.NET Core MVC web application, identifying and highlighting some of the most important concepts and features on a practical basis. MVC is a well-known three-layered (model, view, and controller) architectural pattern that has simplified the development process

of web applications. It reduces the complexity of the application, and in combination with ASP.NET Core, MVC has proven to offer many features that are useful for building APIs and web-based applications.
In conclusion, the use of the MVC architecture allowed us to build a scalable and modular application that separates the concerns into its components. Separation of concerns makes application testing easier as the relationship between application components is coherent and clear. Also, with this feature, the MVC architecture introduces us to the concept of parallel development as it divides the logic of the application. This project demonstrates the benefits of using design patterns in web development and provides a solid foundation for further development and enhancements in systems with similar functionalities.

References

[1] R. Chaturvedi, S. V. Chande and A. Sharma, "Evaluation and Refinement of MVC Web Application," Journal of Information and Computational Science, 2021.

[2] S. J. Patel and P. D. Pancholi, "Implementation and Comparison of MVC Model in ASP.net Framework," IJRAR- International Journal of Research and Analytical Reviews, pp. 827- 835, 2018.

[3] S. Singh, “MVC Framework: A Modern Web Application Development,” International Research Journal of Engineering and Technology (IRJET), vol.VII, no.01, pp.51-55,2020.

[4] A. Majeed and I. Rauf, “MVC Architecture: A Detailed Insight to the Modern Web Applications Development,” Peer Review Journal of Solar & Photoenergy Systems, vol.I, no. 01, pp. 1-7, 2018

[5] D.-P. Pop and A. Altar, "Designing an MVC Model for Rapid Web Application Development," 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, p. 1172 – 1179, 2014

[6] “InterviewBit,” 30 May 2022, [Online]. Available: https://www.interviewbit.com/blog /mvcarchitecture/ . [Accessed February 2023].

[7] “Microsoft,” 21 February 2023, [Online]. Available: https://learn.microsoft.com/enus/dotnet/ architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps. [Accessed 26 February 2023].