Atlas Music Store Professional

Prezentare proiect curs .NET Development
Scoala Informala de IT -2021

Cursant: Botoroaga Horatiu-Dumitru
Mentori: Daniel Hunyadi, Viorel Bota

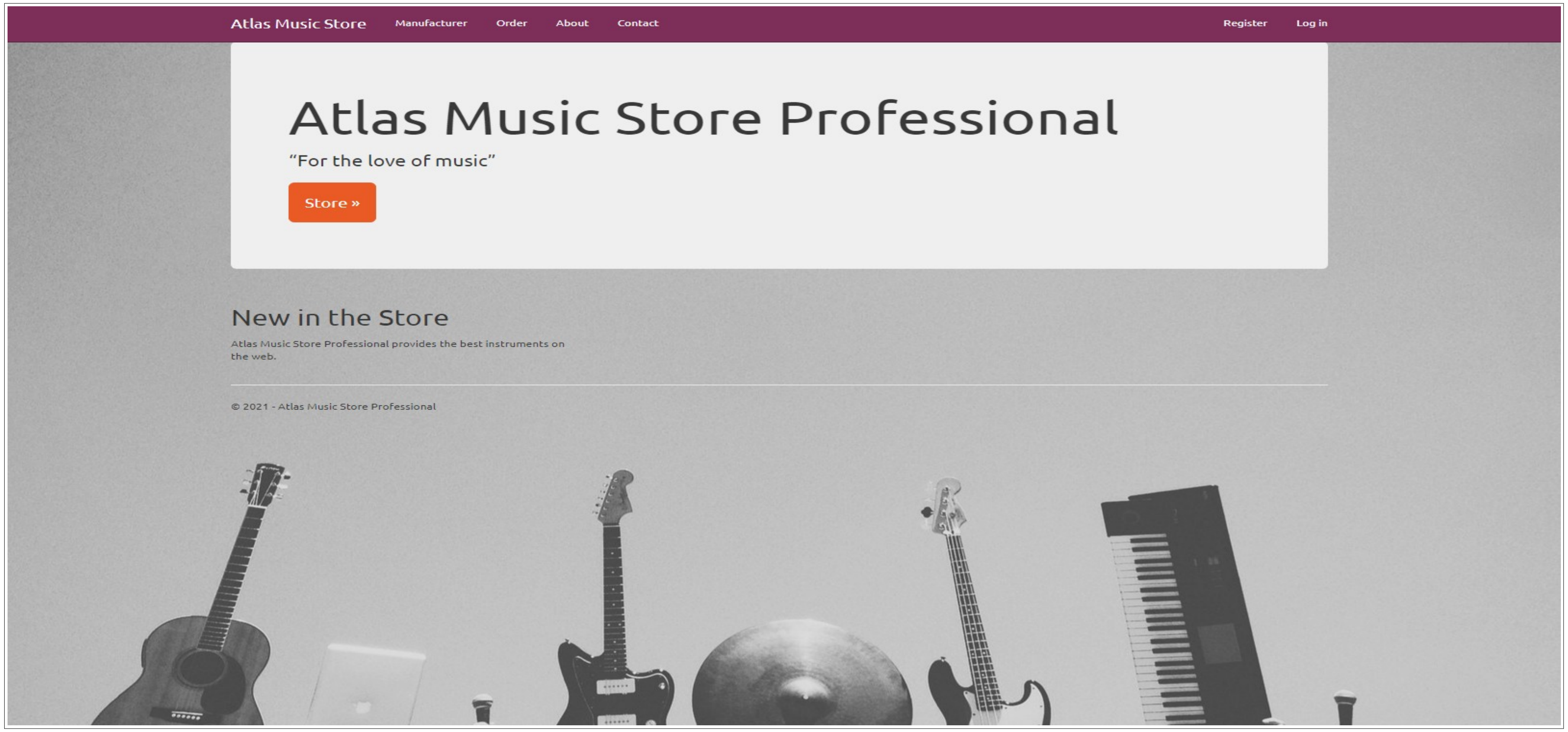
Cuprins:

Prezentare aplicatie

Atlas Music Store Professional este o aplicatie web care isi propune sa vina in ajutorul utilizatorului care doreste sa cumpere instrumente muzicale ale unor branduri cunoscute. Aplicatia poate fi utilizata de administrator si user inregistrati.

In calitate de utilizator, poti sa vizualizezi magazinul cu instrumente muzicale, poti vedea brandurile vandute in web store si comenzile utilizatorului.

In calitate de administrator, poti adauga instrumente in baza de date, edita si modifica descrierea instrumentelor, poti vizualiza userii logati, se pot vedea comenzile tuturor userilor, care pot si modificate sau editate dupa caz.



Atlas Music Store Professional

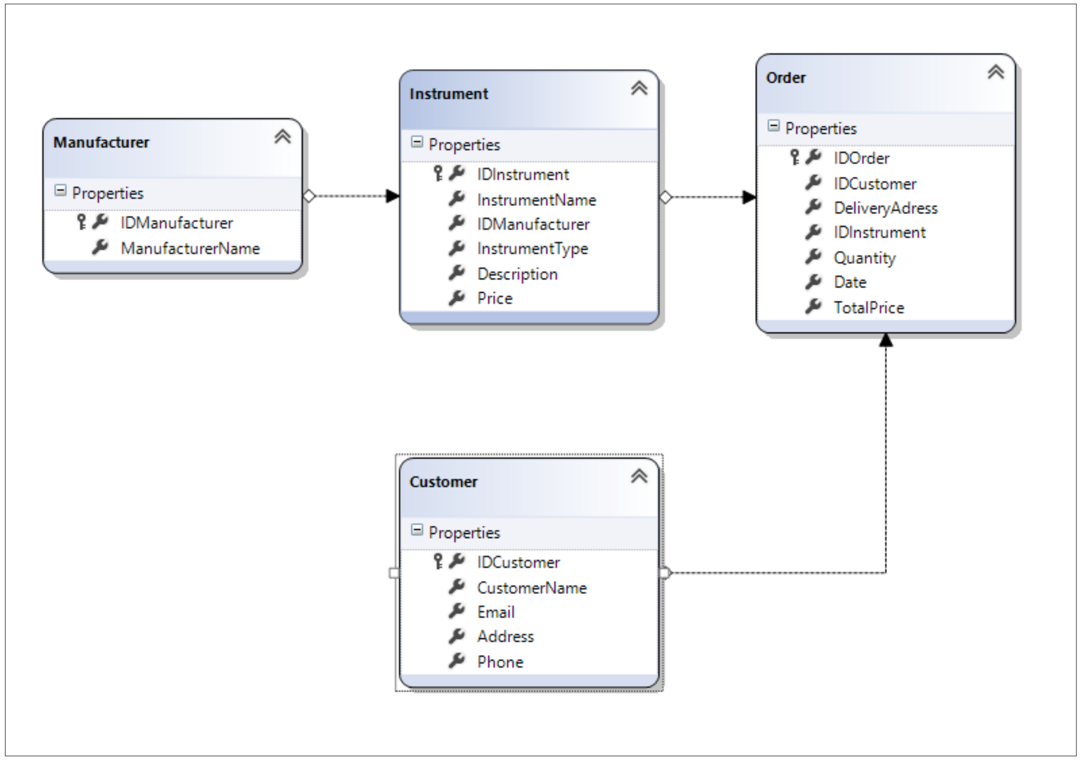
“For the love of music”

New in the Store

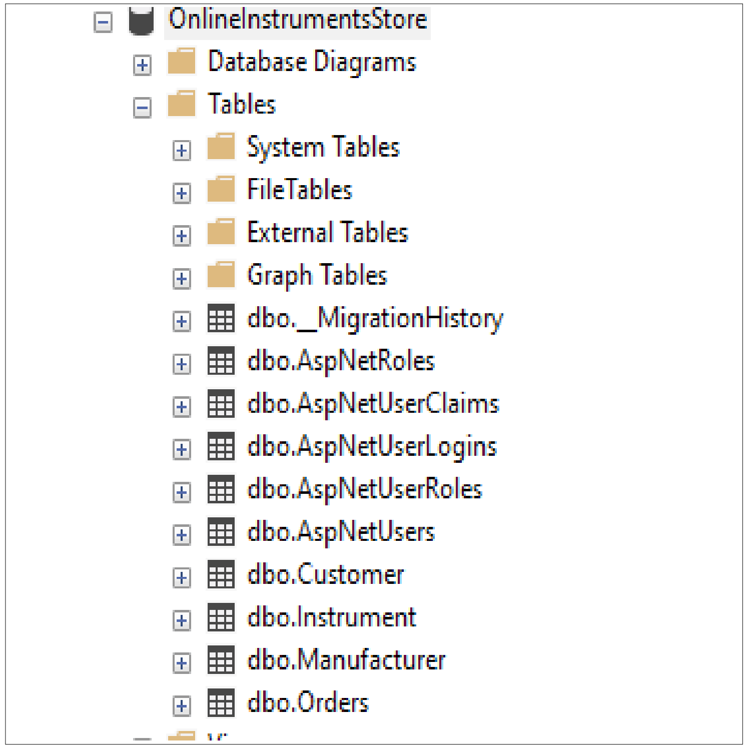
Atlas Music Store Professional provides the best instruments on the web.

© 2021 - Atlas Music Store Professional

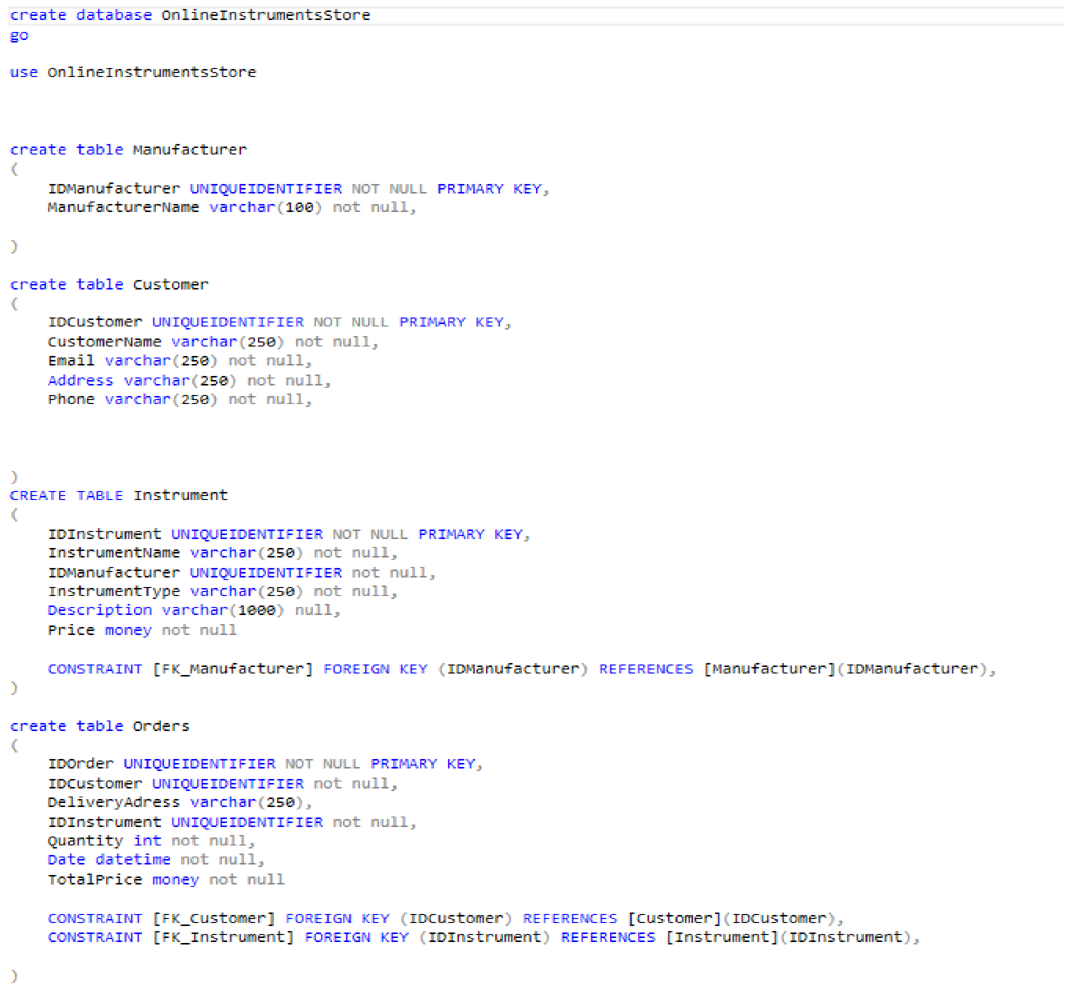
Structura bazei de date



Baza de date este structurata in 4 tabele si anume: Manufacturer (pentru stocare producatorilor de instrumente muzicale), Instrument (pentru stocare datelor referitoare la instrumentele din baza de date), Order (pentru stocare comenzilor fiecarui utilizator), Customer (pentru stocare date referitoare la userii inregistrati in baza de date).



```
OnlineInstrumentsStore
    Database Diagrams
    Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo._MigrationHistory
        dbo.AspNetRoles
        dbo.AspNetUserClaims
        dbo.AspNetUserLogins
        dbo.AspNetUserRoles
        dbo.AspNetUsers
        dbo.Customer
        dbo.Instrument
        dbo.Manufacturer
        dbo.Orders
```

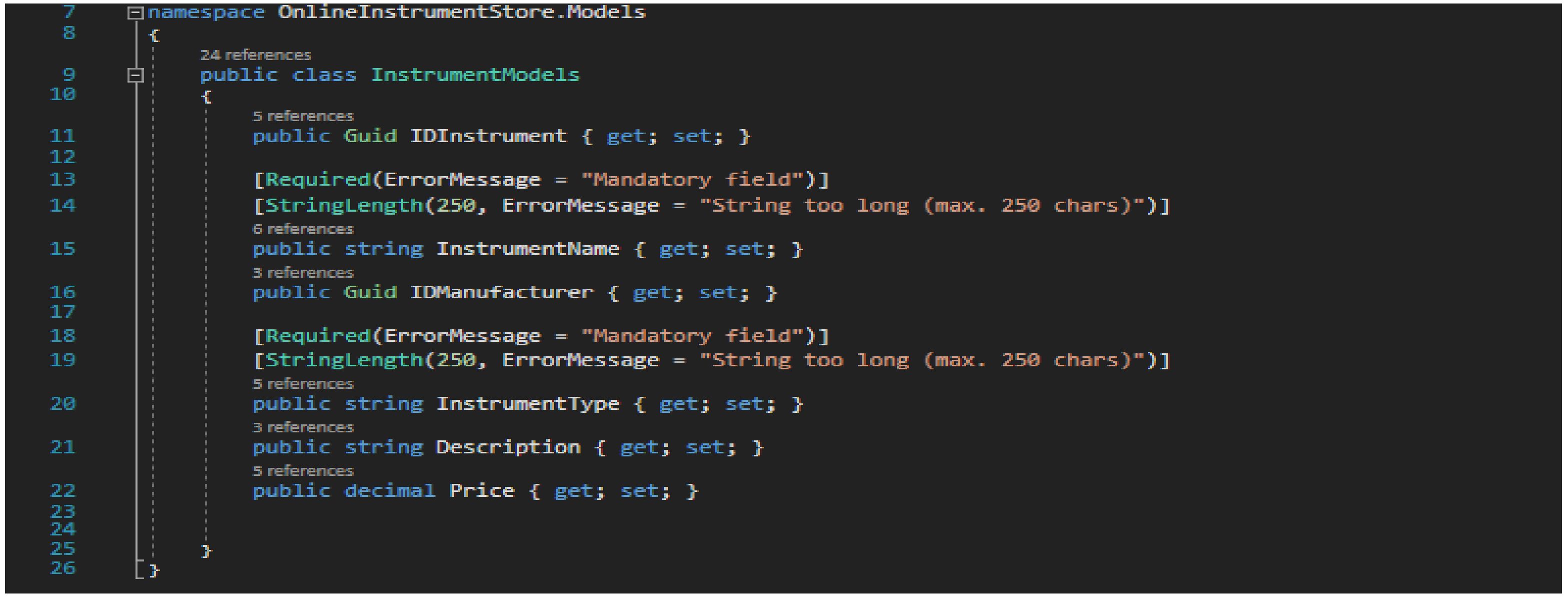


```sql
create database OnlineInstrumentsStore
go

use OnlineInstrumentsStore


create table Manufacturer
(
    IDManufacturer UNIQUEIDENTIFIER NOT NULL PRIMARY KEY,
    ManufacturerName varchar(100) not null,

)

create table Customer
(
    IDCustomer UNIQUEIDENTIFIER NOT NULL PRIMARY KEY,
    CustomerName varchar(250) not null,
    Email varchar(250) not null,
    Address varchar(250) not null,
    Phone varchar(250) not null,



)
CREATE TABLE Instrument
(
    IDInstrument UNIQUEIDENTIFIER NOT NULL PRIMARY KEY,
    InstrumentName varchar(250) not null,
    IDManufacturer UNIQUEIDENTIFIER not null,
    InstrumentType varchar(250) not null,
    Description varchar(1000) null,
    Price money not null

    CONSTRAINT [FK_Manufacturer] FOREIGN KEY (IDManufacturer) REFERENCES [Manufacturer](IDManufacturer),
)

create table Orders
(
    IDOrder UNIQUEIDENTIFIER NOT NULL PRIMARY KEY,
    IDCustomer UNIQUEIDENTIFIER not null,
    DeliveryAdress varchar(250),
    IDInstrument UNIQUEIDENTIFIER not null,
    Quantity int not null,
    Date datetime not null,
    TotalPrice money not null

    CONSTRAINT [FK_Customer] FOREIGN KEY (IDCustomer) REFERENCES [Customer](IDCustomer),
    CONSTRAINT [FK_Instrument] FOREIGN KEY (IDInstrument) REFERENCES [Instrument](IDInstrument),

)
```

Stocarea datelor din aplicatie este realizata prin intermediul unei baze de date tip SQL - platforma Microsoft SQL Server, accesul fiind realizat pe baza de conexiune securizata (eng: integrated security connection string).

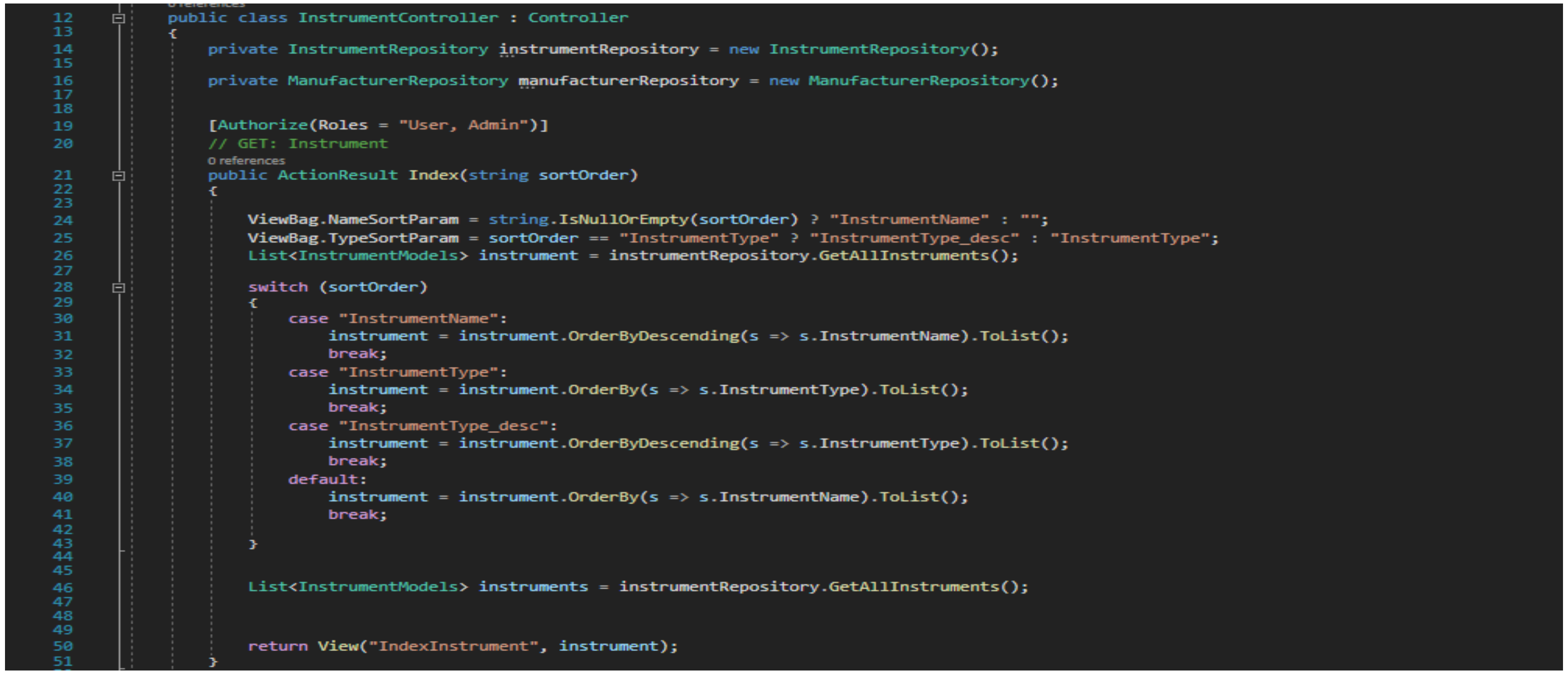
Extrase cod sursa

Instrument Model



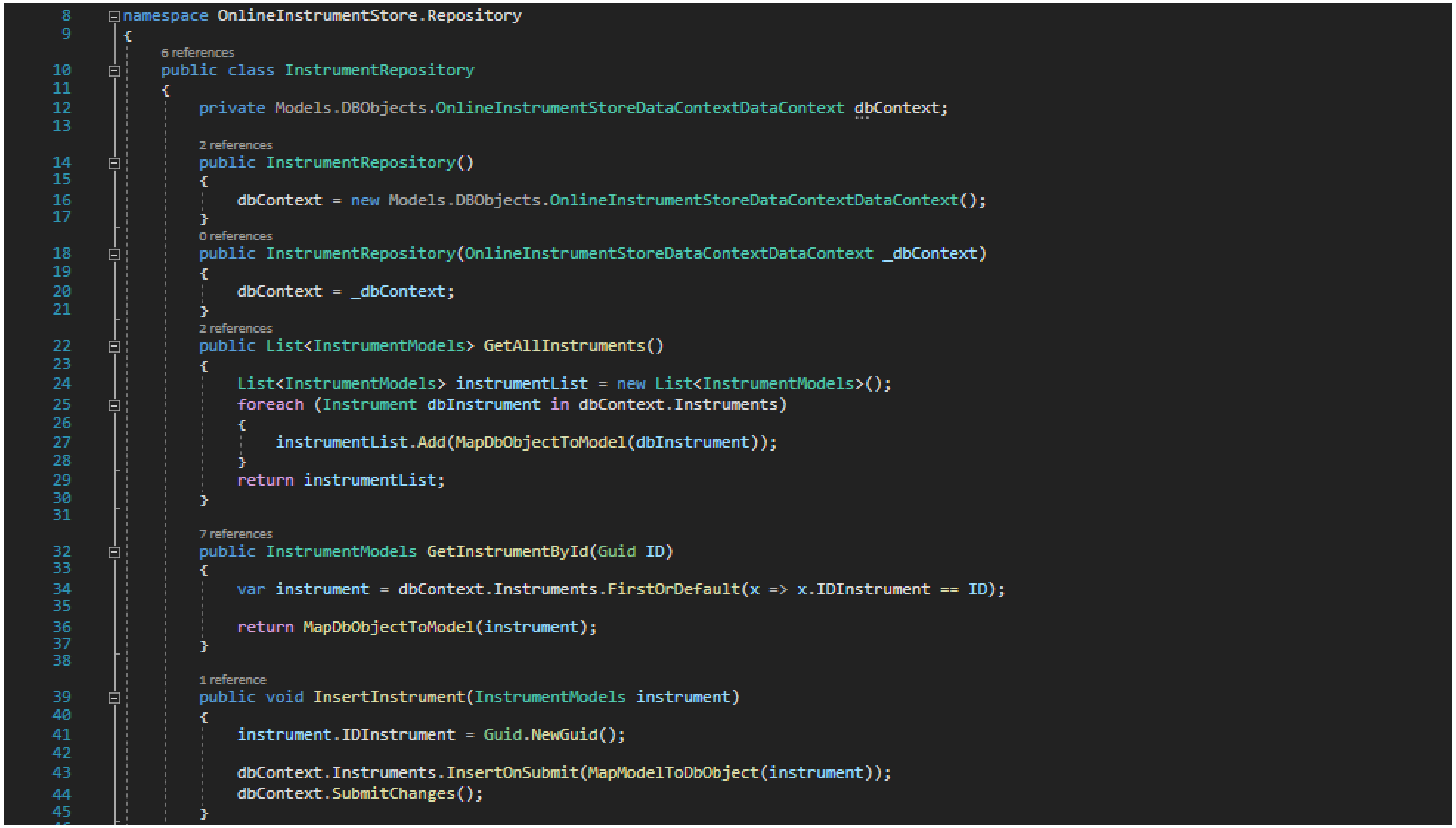
```
namespace OnlineInstrumentStore.Models
{
    24 references
    public class InstrumentModels
    {
        5 references
        public Guid IDInstrument { get; set; }

        [Required(ErrorMessage = "Mandatory field")]
        [StringLength(250, ErrorMessage = "String too long (max. 250 chars)")]
        6 references
        public string InstrumentName { get; set; }
        3 references
        public Guid IDManufacturer { get; set; }

        [Required(ErrorMessage = "Mandatory field")]
        [StringLength(250, ErrorMessage = "String too long (max. 250 chars)")]
        5 references
        public string InstrumentType { get; set; }
        3 references
        public string Description { get; set; }
        5 references
        public decimal Price { get; set; }


    }
}
```

Instrument Controller



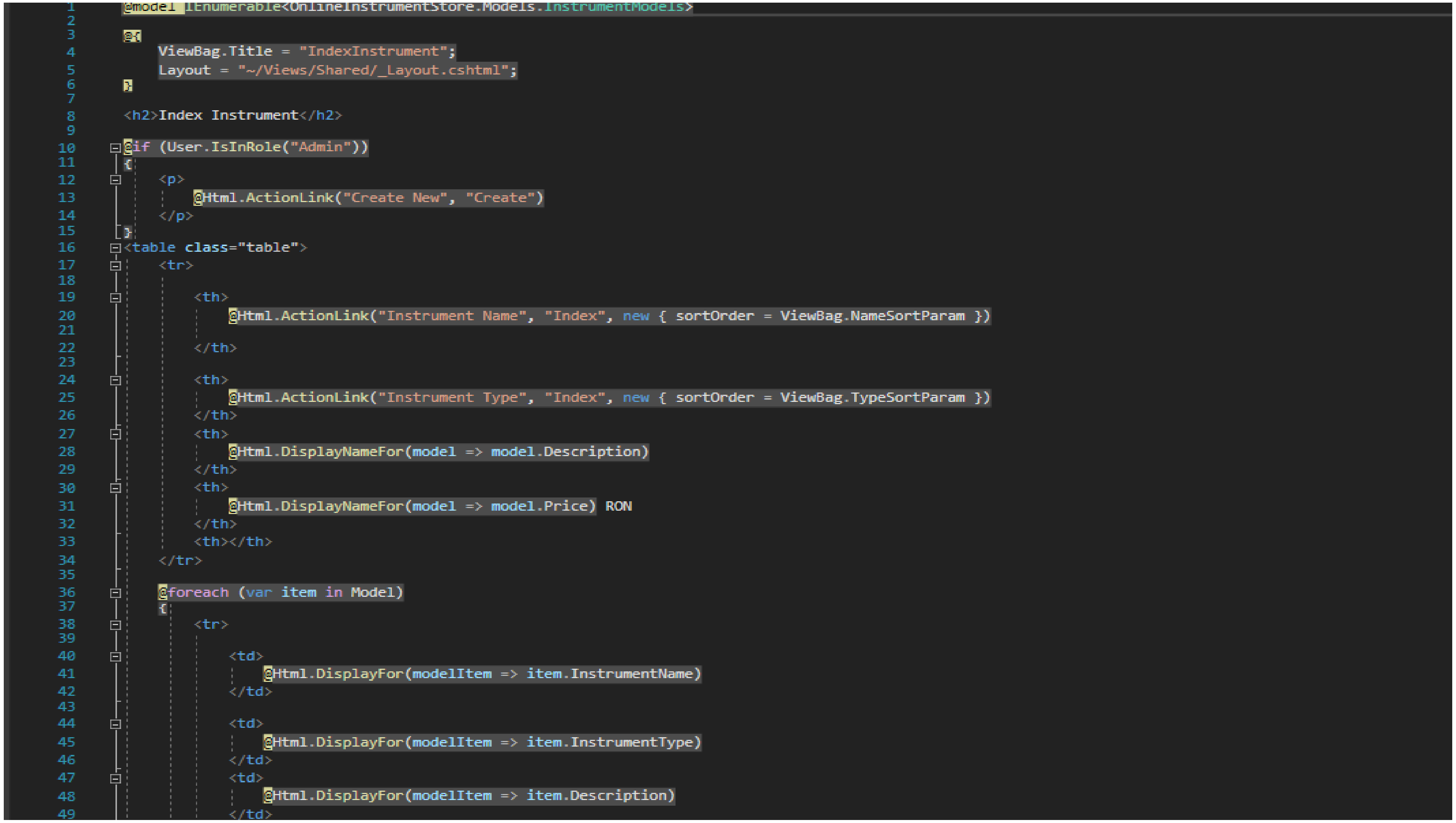
```
    public class InstrumentController : Controller
    {
        private InstrumentRepository instrumentRepository = new InstrumentRepository();

        private ManufacturerRepository manufacturerRepository = new ManufacturerRepository();


        [Authorize(Roles = "User, Admin")]
        // GET: Instrument
        0 references
        public ActionResult Index(string sortOrder)
        {

            ViewBag.NameSortParam = string.IsNullOrEmpty(sortOrder) ? "InstrumentName" : "";
            ViewBag.TypeSortParam = sortOrder == "InstrumentType" ? "InstrumentType_desc" : "InstrumentType";
            List<InstrumentModels> instrument = instrumentRepository.GetAllInstruments();

            switch (sortOrder)
            {
                case "InstrumentName":
                    instrument = instrument.OrderByDescending(s => s.InstrumentName).ToList();
                    break;
                case "InstrumentType":
                    instrument = instrument.OrderBy(s => s.InstrumentType).ToList();
                    break;
                case "InstrumentType_desc":
                    instrument = instrument.OrderByDescending(s => s.InstrumentType).ToList();
                    break;
                default:
                    instrument = instrument.OrderBy(s => s.InstrumentName).ToList();
                    break;

            }


            List<InstrumentModels> instruments = instrumentRepository.GetAllInstruments();



            return View("IndexInstrument", instrument);
        }
```

Instrument Repository



```
namespace OnlineInstrumentStore.Repository
{
    6 references
    public class InstrumentRepository
    {
        private Models.DBObjects.OnlineInstrumentStoreDataContextDataContext dbContext;

        2 references
        public InstrumentRepository()
        {
            dbContext = new Models.DBObjects.OnlineInstrumentStoreDataContextDataContext();
        }
        0 references
        public InstrumentRepository(OnlineInstrumentStoreDataContextDataContext _dbContext)
        {
            dbContext = _dbContext;
        }
        2 references
        public List<InstrumentModels> GetAllInstruments()
        {
            List<InstrumentModels> instrumentList = new List<InstrumentModels>();
            foreach (Instrument dbInstrument in dbContext.Instruments)
            {
                instrumentList.Add(MapDbObjectToModel(dbInstrument));
            }
            return instrumentList;
        }

        7 references
        public InstrumentModels GetInstrumentById(Guid ID)
        {
            var instrument = dbContext.Instruments.FirstOrDefault(x => x.IDInstrument == ID);

            return MapDbObjectToModel(instrument);
        }

        1 reference
        public void InsertInstrument(InstrumentModels instrument)
        {
            instrument.IDInstrument = Guid.NewGuid();

            dbContext.Instruments.InsertOnSubmit(MapModelToDbObject(instrument));
            dbContext.SubmitChanges();
        }
```

Instrument View Edit

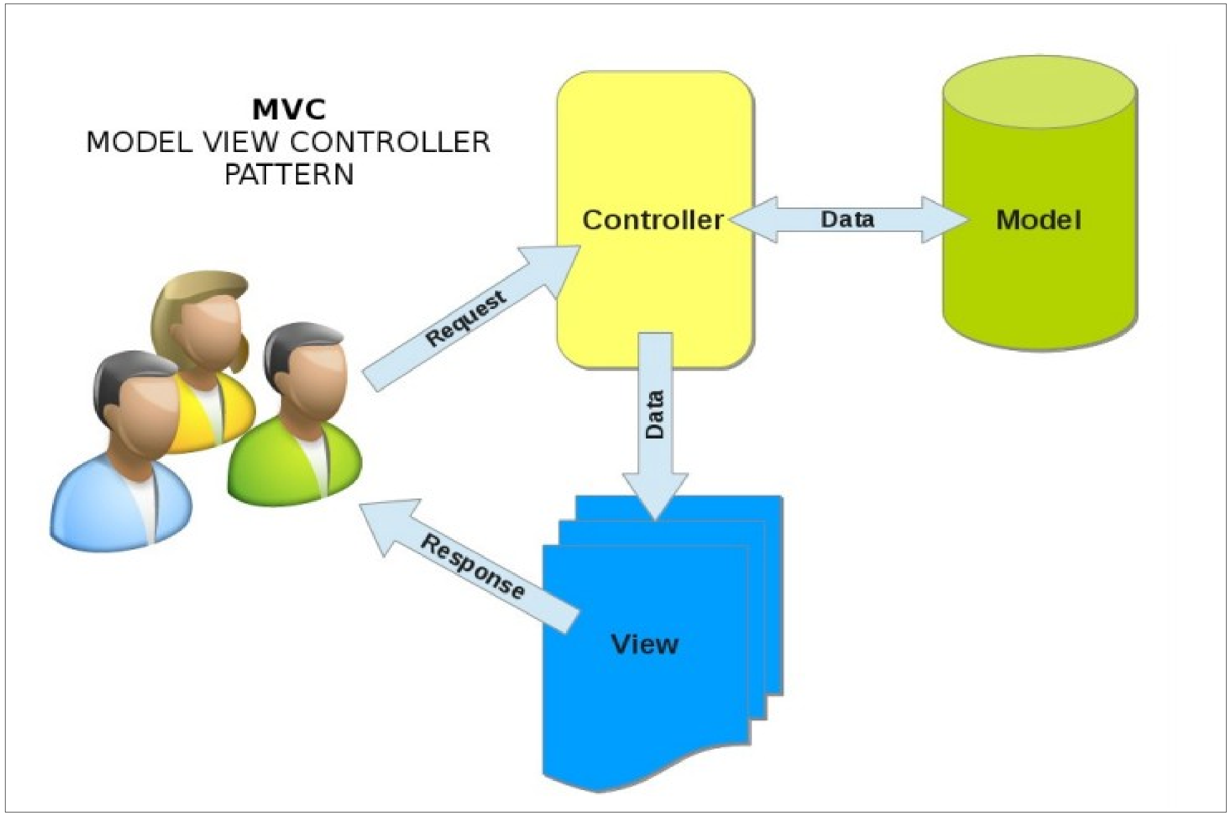


```
@model IEnumerable<OnlineInstrumentStore.Models.InstrumentModels>

@{
    ViewBag.Title = "IndexInstrument";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index Instrument</h2>

@if (User.IsInRole("Admin"))
{
    <p>
        @Html.ActionLink("Create New", "Create")
    </p>
}
<table class="table">
    <tr>

        <th>
            @Html.ActionLink("Instrument Name", "Index", new { sortOrder = ViewBag.NameSortParam })

        </th>

        <th>
            @Html.ActionLink("Instrument Type", "Index", new { sortOrder = ViewBag.TypeSortParam })
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Description)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price) RON
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>

            <td>
                @Html.DisplayFor(modelItem => item.InstrumentName)
            </td>

            <td>
                @Html.DisplayFor(modelItem => item.InstrumentType)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Description)
            </td>
```

Tehnologii utilizate

ASP.NET Web Application

Aplicatia Atlas Music Store Professional este de tip ASP.NET Web dezvoltata in programul Microsoft Visual Studio Community 2019 - .NET Framework. Limbajul folosit este C#.



MVC (Model – View - Controller) este un model de arhitectura utilizat in ingineria software.

Succesul modelului se datoreaza izolarii partii logice (business) de consideratele interfete cu utilizatorul, rezultand o aplicatie unde aspectul vizual sau/si nivelele inferioare ale regulilor de business sunt mai usor de modificat, fara a afecta alte nivele.

MVC este un model de arhitectura software care separa reprezentarea informatiilor din interactiunea cu utilizatorul cu informatiile in sine.

Modelul MVC defineste aplicatii web cu 3 straturi: • Stratul business (logic) - Model • Afisarea – View • Control de intrare – Controller

SQL

Am folosit acest limbaj de interogare structurat, care vine de la Structured Query Language, pentru ca este o colectie organizata de informatii sau de date structurate, stocate electronic intr-un computer.

La implementarea user interface am folosit urmatoarele tehnologii: HTML, CSS, Javascript, Jquery, Bootstrap.

Alte instrumente folosite: GitHub - Remote repository

Bibliografie

- docs.microsoft.com.
- Cursuri – Scoala Informala de IT.
- Youtube.
- Wikipedia.