Dokumentacja projektu z Programowania Zaawansowego – „System do zarządzania magazynem muzycznym”.

Przygotował Kacper Trocewicz

Nr. Albumu: 88639

Uniwersytet w Siedlcach

Data: 15.06.2025

Prowadzący: mgr Łukasz Marchel

Spis treści

[2 Link do repozytorium 4](#_Toc200908170)

[3 Schemat Bazy danych 4](#_Toc200908171)

[4 Klasy pomocnicze 4](#_Toc200908172)

[4.1 PaginatedList 4](#_Toc200908173)

[4.2 SeedData – Klasa służąca do wstawienia przykładowych rekordów do bazy danych 5](#_Toc200908174)

[4.3 DocumentType – Enum określający typy dokumentów 5](#_Toc200908175)

[5 Modele 5](#_Toc200908176)

[5.1 Address – zawiera adresy dostawców 5](#_Toc200908177)

[5.2 Category – kategorie instrumentów 6](#_Toc200908178)

[5.3 Document – podstawowe dane dokumentu 6](#_Toc200908179)

[5.4 DocumentInstrument – pozycje z instrumentami na dokumencie 7](#_Toc200908180)

[5.5 ErrorViewModel – model do obsługi błędów 7](#_Toc200908181)

[5.6 FeatureDefinition – definicje cech instrumentów 7](#_Toc200908182)

[5.7 Instrument – model obrazujący instrument na magazynie 8](#_Toc200908183)

[5.8 InstrumentFeature – Pozycje cech danego instrumentu 9](#_Toc200908184)

[5.9 InstrumentInventory – instrument w magazynie 9](#_Toc200908185)

[5.10 Supplier – dostawca instrumentów 10](#_Toc200908186)

[6 Kontrolery 10](#_Toc200908187)

[6.1 DocumentsController – służy do osbuługi dokumentów, które wpływają na stan magazynowy. 10](#_Toc200908188)

[6.2 HomeController – obsługa strony głównej 14](#_Toc200908189)

[6.3 InstrumentFeaturesController – edycja cech danego instrumentu obecnie nie obsłużona przez interfejs użytkownika ale w przyszłości będzie to bardzo ważna funckaj 15](#_Toc200908190)

[6.4 InstrumentInventoriesController – zarządzanie stanem magazynowym, czyli ilością oraz tylko tymi instrumentami które zostały przyjęte do magazynu 18](#_Toc200908191)

[6.5 InstrumentsController – crud dla istrumentów w ogólnej sieci dystrybucji 24](#_Toc200908192)

[6.6 SuppliersController – zarządzanie dostawcami instrumentów 29](#_Toc200908193)

[7 Widoki 33](#_Toc200908194)

[7.1 Documents 33](#_Toc200908195)

[7.1.1 Create – utworzenie dokumentu na podstawie dostawcy i listy instrumentów od tego dostawcy 33](#_Toc200908196)

[7.1.2 Delete – usunięcie dokumentu 34](#_Toc200908197)

[7.1.3 Details – szczegóły dokumentu, wraz z danymi o instrumentach które są jako pozycje dokumentu i podsumowaniami ilości i wartości pozycji na dokumencie 35](#_Toc200908198)

[7.1.4 Edit – edycja dokumentu 37](#_Toc200908199)

[7.1.5 Index – wyświetlenie wszystkich dokumentów 37](#_Toc200908200)

[7.2 Home 38](#_Toc200908201)

[7.2.1 About – opis aplikacji 38](#_Toc200908202)

[7.2.2 Privacy – strona dotycząca prywatność na stronie 38](#_Toc200908203)

[7.2.3 Index – widok strony głównej 39](#_Toc200908204)

[7.3 InstrumentFeatures 40](#_Toc200908205)

[7.3.1 Create – utworzenie cechy dla danego instrumentu na podstawie definicji cechy. 40](#_Toc200908206)

[7.3.2 Delete – usunięcie cechy danego instrumentu 40](#_Toc200908207)

[7.3.3 Details – szczegóły cechy instrumentu 41](#_Toc200908208)

[7.3.4 Edit – edycja cechy instrumentu 42](#_Toc200908209)

[7.3.5 Index – widok cech instrumentów 43](#_Toc200908210)

[7.4 InstrumentInventories – główna obsługa magazynu czyli najważniejszy widok w projekcie. służy do dodawania instrumentów na podstawie dokumentów przyjęcia zewnętrznego. można w nim zarządzać instrumentami i jego stanami magazynowymi. 44](#_Toc200908211)

[7.4.1 Create – utworzenie nowego przyjęcia magazynowego na podstawie instrumentów od danego dostawcy przy pomocy dokumentu przyjęcia magazynowego 44](#_Toc200908212)

[7.4.2 Delete – usnięcie instrumentu z magazynu 45](#_Toc200908213)

[7.4.3 Details – szczegóły instrumentu z magazynu 46](#_Toc200908214)

[7.4.4 Edit – edycja instrumentów z magazynu 48](#_Toc200908215)

[7.4.5 Index – widok instrumentów które są w magazynie 50](#_Toc200908216)

[7.5 Instruments 53](#_Toc200908217)

[7.5.1 Create – tworzenie isntrumentu w systemie 53](#_Toc200908218)

[7.5.2 Delete 55](#_Toc200908219)

[7.5.3 Details – szczegóły instrumentu 57](#_Toc200908220)

[7.5.4 Edit – edycja instrumentu 59](#_Toc200908221)

[7.5.5 Index – widok wszystkich instrumentów w sieci dystrybucyjnej 62](#_Toc200908222)

[7.6 Suppliers – zarządzanie dostawcami 65](#_Toc200908223)

[7.6.1 Create – tworzenie dostawcy 65](#_Toc200908224)

[7.6.2 Delete 66](#_Toc200908225)

[7.6.3 Details – szczegóły dostawcy 66](#_Toc200908226)

[7.6.4 Edit 67](#_Toc200908227)

[7.6.5 Index – lista dostawców isntrumentów 68](#_Toc200908228)

[7.7 Shared 69](#_Toc200908229)

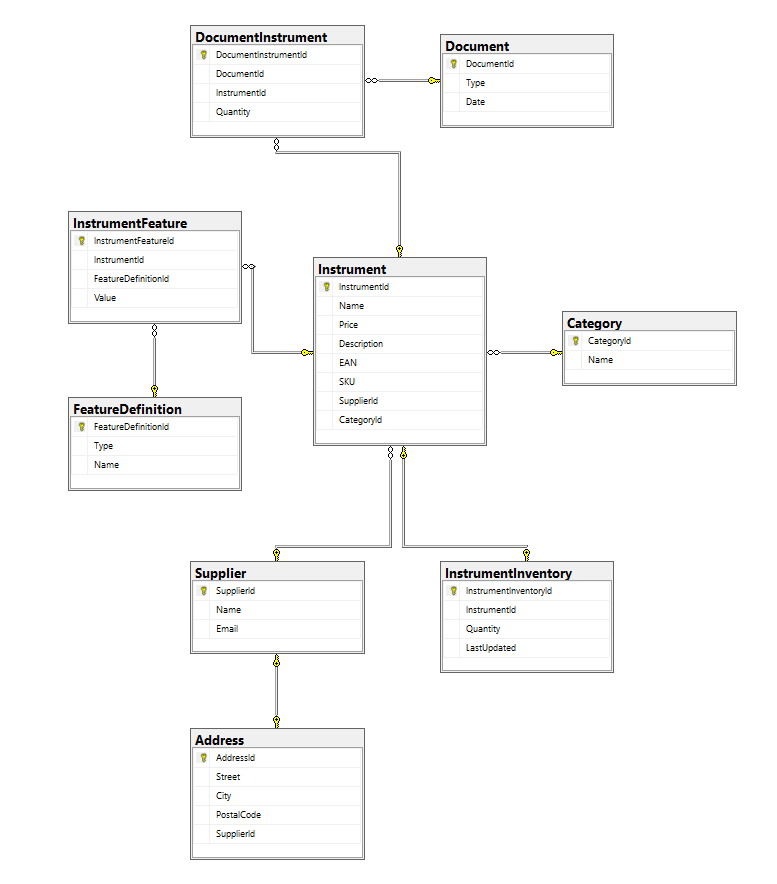
[7.7.1 \_Layout – ogólny widok dla każdej strony 69](#_Toc200908230)

[7.7.2 error – widok błędu 70](#_Toc200908231)

# Link do repozytorium

<https://github.com/trocekbp/Music-Store-Warehouse-App>

# Schemat Bazy danych



# Klasy pomocnicze

## PaginatedList

using Microsoft.EntityFrameworkCore;

namespace Music\_Store\_Warehouse\_App.Data

{

public class PaginatedList<T> : List<T>

{

public int PageIndex { get; private set; }

public int TotalPages { get; private set; }

public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)

{

PageIndex = pageIndex;

TotalPages = (int)Math.Ceiling(count / (double)pageSize);

this.AddRange(items);

}

public bool HasPreviousPage => PageIndex > 1;

public bool HasNextPage => PageIndex < TotalPages;

public static async Task<PaginatedList<T>> CreateAsync(IQueryable<T> source, int pageIndex, int pageSize)

{

var count = await source.CountAsync();

var items = await source.Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToListAsync();

return new PaginatedList<T>(items, count, pageIndex, pageSize);

}

}

}

## SeedData – Klasa służąca do wstawienia przykładowych rekordów do bazy danych

## DocumentType – Enum określający typy dokumentów

namespace Music\_Store\_Warehouse\_App.Models.Enums

{

public enum DocumentType

{

PZ, //Przyjęcie zewnętrzne

WZ, //Wydanie zewnętrzne

ZZ, //Zlecenie Zamówienia na przyszłość

}

}

# Modele

## Address – zawiera adresy dostawców

using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Text.RegularExpressions;

namespace Music\_Store\_Warehouse\_App.Models

{

public class Address

{

public int AddressId { get; set; }

[Required(ErrorMessage = "Ulica jest wymagana.")]

[StringLength(200, ErrorMessage = "Ulica nie może mieć więcej niż 200 znaków.")]

[DisplayName("Ulica")]

public string Street { get; set; }

[Required(ErrorMessage = "Miasto jest wymagane.")]

[StringLength(100, ErrorMessage = "Miasto nie może mieć więcej niż 100 znaków.")]

[DisplayName("Miasto")]

public string City { get; set; }

[Required(ErrorMessage = "Kod pocztowy jest wymagany.")]

[RegularExpression(@"^(?:\d{5}|\d{2}\D\d{3}|\d{3}\D\d{2})$",

ErrorMessage = "Nieprawidłowy format kodu pocztowego (np. 00000, 00-001 PL lub 000 01 CZ).")]

[DisplayName("Kod pocztowy")]

public string PostalCode { get; set; }

// → To jest FK do Supplier.SupplierId:

[ValidateNever]

public int SupplierId { get; set; }

// Nawigacja odwrotna

[ValidateNever]

public Supplier Supplier { get; set; }

}

}

## Category – kategorie instrumentów

using System.ComponentModel;

using System.Diagnostics.Metrics;

namespace Music\_Store\_Warehouse\_App.Models

{

public class Category

{

public int CategoryId { get; set; }

[DisplayName("Nazwa kategorii")]

public string Name { get; set; }

public ICollection<Instrument> Instruments { get; set; }

}

}

## Document – podstawowe dane dokumentu

using Music\_Store\_Warehouse\_App.Models.Enums;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

namespace Music\_Store\_Warehouse\_App.Models

{

public class Document

{

public int DocumentId { get; set; }

[Required(ErrorMessage = "Typ dokumentu jest wymagany !")]

[DisplayName("Typ")]

public DocumentType Type { get; set; }

[DataType(DataType.Date)]

[DisplayName("Data wystawienia")]

public DateTime Date { get; set; } = DateTime.Now; //Dana wystawienia

public ICollection<DocumentInstrument> DocumentInstruments { get; set; }

= new List<DocumentInstrument>();

}

}

## DocumentInstrument – pozycje z instrumentami na dokumencie

using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

using Music\_Store\_Warehouse\_App.Models.Enums;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Reflection.Metadata;

namespace Music\_Store\_Warehouse\_App.Models

{

public class DocumentInstrument

{

public int DocumentInstrumentId { get; set; }

public int DocumentId { get; set; }

[ValidateNever]

public Document Document { get; set; }

public int InstrumentId { get; set; }

[ValidateNever]

public Instrument Instrument { get; set; }

[DisplayName("Ilość")]

[Required(ErrorMessage = "Ilość jest wymagana")]

[Range(0, int.MaxValue, ErrorMessage = "Ilość nie może być ujemna")]

public int Quantity { get; set; } // ilość przyjęta / wydana

}

}

## ErrorViewModel – model do obsługi błędów

namespace Music\_Store\_Warehouse\_App.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

## FeatureDefinition – definicje cech instrumentów

using System.ComponentModel;

namespace Music\_Store\_Warehouse\_App.Models

{

public enum FType //Feature Type - Typ na podstawie jego dobieramy właściwości dla odpowiednich instrumentów

{

Gitary,

Perkusje,

Pianina,

Skrzypce,

Dęte,

Inne //narazie inne są nieobsłużone

}

public class FeatureDefinition

{

public int FeatureDefinitionId { get; set; }

public FType Type { get; set; } //Typ

[DisplayName("Cecha")]

public string Name { get; set; } // np. "Ilość strun"

public ICollection<InstrumentFeature> InstrumentFeatures { get; set; }

}

}

## Instrument – model obrazujący instrument na magazynie

using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

using Newtonsoft.Json.Serialization;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

namespace Music\_Store\_Warehouse\_App.Models

{

public class Instrument

{

public int InstrumentId { get; set; }

[DisplayName("Nazwa")]

public string Name { get; set; }

[DisplayName("Cena")]

[Required(ErrorMessage = "Cena jest wymagana")]

[RegularExpression(@"^\d+([.,]\d{1,2})?$",

ErrorMessage = "Podaj cenę w formacie liczby całkowitej lub z maksymalnie dwoma cyframi po przecinku (np. 123,45)")]

[DataType(DataType.Currency, ErrorMessage = "Podaj w formacie waluty")]

public decimal Price { get; set; }

[DisplayName("Opis")]

public string Description { get; set; }

// Kod kreskowy – do skanowania lub importu z producenta

[DisplayName("Kod ean")]

[RegularExpression(@"^\d{13}$", ErrorMessage = "EAN musi zawierać dokładnie 13 znaków")]

public string EAN { get; set; }

// Numer magazynowy – do zarządzania wewnętrznego

[DisplayName("Numer magazynowy")]

[StringLength(20)]

public string SKU { get; set; } //Stock Keeping Unit

// Relacja do dostawcy, może być null we wstępnej fazie projektu

public int? SupplierId { get; set; }

[DisplayName("Dostawca")]

[ValidateNever] //Dzięki temu nie będzie problemu z tworzeniem klasy, WAŻNE

public Supplier? Supplier { get; set; }

// Relacja do kategorii

[Required(ErrorMessage ="Kategoria jest wymagana")]

public int CategoryId { get; set; }

[DisplayName("Kategoria")]

[ValidateNever]

public Category Category { get; set; }

// Nawigacja do przypisanych cech

public ICollection<InstrumentFeature>? InstrumentFeatures { get; set; }

//Nawigacja do stanu na magazynie

[ValidateNever]

public InstrumentInventory Inventory { get; set; }

}

}

## InstrumentFeature – Pozycje cech danego instrumentu

using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

using System.ComponentModel;

namespace Music\_Store\_Warehouse\_App.Models

{

public class InstrumentFeature

{

public int InstrumentFeatureId { get; set; }

public int InstrumentId { get; set; }

[ValidateNever]

public Instrument Instrument { get; set; }

public int FeatureDefinitionId { get; set; }

[ValidateNever]

public FeatureDefinition FeatureDefinition { get; set; }

public string Value { get; set; } // np. "Olcha"

}

}

## InstrumentInventory – instrument w magazynie

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel;

namespace Music\_Store\_Warehouse\_App.Models

{

public class InstrumentInventory

{

public int InstrumentInventoryId { get; set; }

[Required(ErrorMessage = "Instrument jest wymagany !")]

public int InstrumentId { get; set; }

public Instrument Instrument { get; set; }

[DisplayName("Ilość")]

[Required(ErrorMessage = "Ilość jest wymagana")]

[Range(0, int.MaxValue, ErrorMessage = "Ilość nie może być ujemna")]

public int Quantity { get; set; }

public DateTime LastUpdated { get; set; } = DateTime.UtcNow;

}

}

## Supplier – dostawca instrumentów

using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Collections.Generic;

namespace Music\_Store\_Warehouse\_App.Models

{

public class Supplier

{

public int SupplierId { get; set; }

[Required(ErrorMessage = "Nazwa dostawcy jest wymagana.")]

[StringLength(100, ErrorMessage = "Nazwa nie może mieć więcej niż 100 znaków.")]

[DisplayName("Nazwa")]

public string Name { get; set; }

[EmailAddress(ErrorMessage = "Nieprawidłowy format adresu e-mail.")]

[DisplayName("E-mail")]

public string Email { get; set; }

// Nawigacja

public Address Address { get; set; }

// Relacja 1:n — Dostawca ma wiele instrumentów

[ValidateNever]

public ICollection<Instrument> Instruments { get; set; }

}

}

# Kontrolery

## DocumentsController – służy do osbuługi dokumentów, które wpływają na stan magazynowy.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Music\_Store\_Warehouse\_App.Data;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers

{

public class DocumentsController : Controller

{

private readonly Music\_Store\_Warehouse\_AppContext \_context;

public DocumentsController(Music\_Store\_Warehouse\_AppContext context)

{

\_context = context;

}

// GET: Documents

public async Task<IActionResult> Index()

{

return View(await \_context.Document

.Include(i => i.DocumentInstruments)

.ThenInclude(instr => instr.Instrument)

.OrderByDescending(doc => doc.Date)

.ToListAsync());

}

// GET: Documents/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var document = await \_context.Document

.Include(i => i.DocumentInstruments)

.ThenInclude(d => d.Instrument)

.ThenInclude(c => c.Category)

.Include(i => i.DocumentInstruments)

.ThenInclude(d => d.Instrument)

.ThenInclude(s => s.Supplier)

.FirstOrDefaultAsync(m => m.DocumentId == id);

if (document == null)

{

return NotFound();

}

var supplierId = document.DocumentInstruments

.Select(di => di.Instrument?.Supplier?.SupplierId)

.FirstOrDefault();

var supplier = await \_context.Supplier

.Include(s => s.Address)

.FirstOrDefaultAsync(s => s.SupplierId == supplierId);

ViewBag.Supplier = supplier;

return View(document);

}

// GET: Documents/Create?supplierId=5&selectedInstrumentIds=2&selectedInstrumentIds=7

public IActionResult Create(int? supplierId, List<int> selectedInstrumentIds)

{

var supplier = \_context.Supplier

.Include(i => i.Address)

.FirstOrDefault(i => i.SupplierId == supplierId);

if (supplier == null)

return NotFound();

var instrument\_list = \_context.Instrument

.Where(i => selectedInstrumentIds.Contains(i.InstrumentId))

.Include(i => i.Category)

.Include(i => i.Supplier)

.ToList();

var documentItems = new List<DocumentInstrument>();

foreach(var item in instrument\_list) {

documentItems.Add(new DocumentInstrument

{

Instrument = item,

InstrumentId = item.InstrumentId

});

}

var document = new Document()

{

Type = Models.Enums.DocumentType.PZ,

DocumentInstruments = documentItems

};

//Opcjonalnie przekazujemy do ViewBag dodatkowe dane (np. nazwa dostawcy)

ViewBag.Supplier = supplier;

return View(document);

}

// POST: Documents/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("DocumentId, Type")] Document document,

[FromForm] IList<DocumentInstrument> documentInstruments)

{

document.DocumentInstruments = documentInstruments;

if (ModelState.IsValid)

{

document.Date = DateTime.Now;

//Aktualizacja stanu

foreach (var item in document.DocumentInstruments) {

var instrumentInventory = \_context.InstrumentInventory

.FirstOrDefault(i => i.InstrumentId == item.InstrumentId);

if (instrumentInventory == null)

{

\_context.InstrumentInventory.Add(new InstrumentInventory() //Jeśli jeszcze nie ma takiego instrumentu w magazynie

{

InstrumentId = item.InstrumentId,

Quantity = item.Quantity

});

}

else {

instrumentInventory.Quantity += item.Quantity;

}

}

\_context.Add(document);

//Aktualizacja stanu magazynowego

// var inventory = \_context.

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(document);

}

// GET: Documents/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var document = await \_context.Document.FindAsync(id);

if (document == null)

{

return NotFound();

}

return View(document);

}

// POST: Documents/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("DocumentId,Type,Date")] Document document)

{

if (id != document.DocumentId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(document);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!DocumentExists(document.DocumentId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(document);

}

// GET: Documents/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var document = await \_context.Document

.FirstOrDefaultAsync(m => m.DocumentId == id);

if (document == null)

{

return NotFound();

}

return View(document);

}

// POST: Documents/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var document = await \_context.Document.FindAsync(id);

if (document != null)

{

\_context.Document.Remove(document);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool DocumentExists(int id)

{

return \_context.Document.Any(e => e.DocumentId == id);

}

}

}

## HomeController – obsługa strony głównej

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers;

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

public IActionResult About()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

## InstrumentFeaturesController – edycja cech danego instrumentu obecnie nie obsłużona przez interfejs użytkownika ale w przyszłości będzie to bardzo ważna funckaj

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Music\_Store\_Warehouse\_App.Data;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers

{

public class InstrumentFeaturesController : Controller

{

private readonly Music\_Store\_Warehouse\_AppContext \_context;

public InstrumentFeaturesController(Music\_Store\_Warehouse\_AppContext context)

{

\_context = context;

}

// GET: InstrumentFeatures

public async Task<IActionResult> Index()

{

var music\_Store\_Warehouse\_AppContext = \_context.InstrumentFeature.Include(i => i.FeatureDefinition).Include(i => i.Instrument);

return View(await music\_Store\_Warehouse\_AppContext.ToListAsync());

}

// GET: InstrumentFeatures/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var instrumentFeature = await \_context.InstrumentFeature

.Include(i => i.FeatureDefinition)

.Include(i => i.Instrument)

.FirstOrDefaultAsync(m => m.InstrumentFeatureId == id);

if (instrumentFeature == null)

{

return NotFound();

}

return View(instrumentFeature);

}

// GET: InstrumentFeatures/Create

public IActionResult Create()

{

ViewData["FeatureDefinitionId"] = new SelectList(\_context.FeatureDefinition, "FeatureDefinitionId", "FeatureDefinitionId");

ViewData["InstrumentId"] = new SelectList(\_context.Instrument, "InstrumentId", "InstrumentId");

return View();

}

// POST: InstrumentFeatures/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("InstrumentFeatureId,InstrumentId,FeatureDefinitionId,Value")] InstrumentFeature instrumentFeature)

{

if (ModelState.IsValid)

{

\_context.Add(instrumentFeature);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["FeatureDefinitionId"] = new SelectList(\_context.FeatureDefinition, "FeatureDefinitionId", "FeatureDefinitionId", instrumentFeature.FeatureDefinitionId);

ViewData["InstrumentId"] = new SelectList(\_context.Instrument, "InstrumentId", "InstrumentId", instrumentFeature.InstrumentId);

return View(instrumentFeature);

}

// GET: InstrumentFeatures/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var instrumentFeature = await \_context.InstrumentFeature.FindAsync(id);

if (instrumentFeature == null)

{

return NotFound();

}

ViewData["FeatureDefinitionId"] = new SelectList(\_context.FeatureDefinition, "FeatureDefinitionId", "FeatureDefinitionId", instrumentFeature.FeatureDefinitionId);

ViewData["InstrumentId"] = new SelectList(\_context.Instrument, "InstrumentId", "InstrumentId", instrumentFeature.InstrumentId);

return View(instrumentFeature);

}

// POST: InstrumentFeatures/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("InstrumentFeatureId,InstrumentId,FeatureDefinitionId,Value")] InstrumentFeature instrumentFeature)

{

if (id != instrumentFeature.InstrumentFeatureId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(instrumentFeature);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!InstrumentFeatureExists(instrumentFeature.InstrumentFeatureId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["FeatureDefinitionId"] = new SelectList(\_context.FeatureDefinition, "FeatureDefinitionId", "FeatureDefinitionId", instrumentFeature.FeatureDefinitionId);

ViewData["InstrumentId"] = new SelectList(\_context.Instrument, "InstrumentId", "InstrumentId", instrumentFeature.InstrumentId);

return View(instrumentFeature);

}

// GET: InstrumentFeatures/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var instrumentFeature = await \_context.InstrumentFeature

.Include(i => i.FeatureDefinition)

.Include(i => i.Instrument)

.FirstOrDefaultAsync(m => m.InstrumentFeatureId == id);

if (instrumentFeature == null)

{

return NotFound();

}

return View(instrumentFeature);

}

// POST: InstrumentFeatures/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var instrumentFeature = await \_context.InstrumentFeature.FindAsync(id);

if (instrumentFeature != null)

{

\_context.InstrumentFeature.Remove(instrumentFeature);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool InstrumentFeatureExists(int id)

{

return \_context.InstrumentFeature.Any(e => e.InstrumentFeatureId == id);

}

}

}

## InstrumentInventoriesController – zarządzanie stanem magazynowym, czyli ilością oraz tylko tymi instrumentami które zostały przyjęte do magazynu

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Music\_Store\_Warehouse\_App.Data;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers

{

public class InstrumentInventoriesController : Controller

{

private readonly Music\_Store\_Warehouse\_AppContext \_context;

public InstrumentInventoriesController(Music\_Store\_Warehouse\_AppContext context)

{

\_context = context;

}

// GET: InstrumentInventories

//Klasa ta różnie się tym że operujemy tlyko na instrumentach które są na magazynie poprzez .Where(i => i.inventory != null)

public async Task<IActionResult> Index(

string sortOrder,

string currentFilter,

string searchString,

int? pageNumber,

int? categoryId,

int? supplierId)

{

ViewData["CurrentSort"] = sortOrder;

ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ? "name\_desc" : "";

ViewData["PriceSortParm"] = sortOrder == "Price" ? "price\_desc" : "Price";

ViewData["CurrentFilter"] = searchString;

if (searchString != null)

{

pageNumber = 1;

}

else

{

searchString = currentFilter;

}

var categories = await \_context.Category

.OrderBy(c => c.Name)

.ToListAsync();

var suppliers = await \_context.Supplier

.OrderBy(s => s.Name)

.ToListAsync();

ViewData["CategoryList"] = new SelectList(categories, "CategoryId", "Name", categoryId);

ViewData["CurrentCategory"] = categoryId; // by wiedzieć, która opcja ma być selected

ViewData["SupplierList"] = new SelectList(suppliers, "SupplierId", "Name", supplierId);

ViewData["CurrentSupplier"] = supplierId; // by wiedzieć, która opcja ma być selected

IQueryable<Instrument> instruments = \_context.Instrument // instruments = INSTRUMENTS IN INVENTORY !!!

.Include(i => i.Category)

.Include(i => i.Inventory)

.Include(i => i.Supplier)

.Where(i => i.Inventory != null); //Czyli żeby było prościej pobieramy tylko instrumenty które są na stanie

// --- Filtrowanie po wyszukiwanej frazie ---

if (!String.IsNullOrEmpty(searchString))

{

instruments = instruments.Where(s =>

s.Name.Contains(searchString) ||

s.Supplier.Name.Contains(searchString));

}

// --- Filtrowanie po kategorii, jeśli użytkownik wybrał categoryId ---

if (categoryId.HasValue)

{

instruments = instruments.Where(i => i.CategoryId == categoryId.Value);

}

if (supplierId.HasValue)

{

instruments = instruments.Where(i => i.SupplierId == supplierId.Value);

}

switch (sortOrder)

{

case "name\_desc":

instruments = instruments.OrderByDescending(s => s.Name);

break;

case "Price":

instruments = instruments.OrderBy(s => s.Price);

break;

case "price\_desc":

instruments = instruments.OrderByDescending(s => s.Price);

break;

default:

instruments = instruments.OrderBy(s => s.Name);

break;

}

int pageSize = 10;

return View(await PaginatedList<Instrument>.CreateAsync(instruments.AsNoTracking(), pageNumber ?? 1, pageSize));

}

// GET: InstrumentInventories/Details/5

// GET: Instruments/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var instrument = await \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Supplier)

.Include(i => i.Inventory)

.Include(i => i.InstrumentFeatures)

.ThenInclude(ifeat => ifeat.FeatureDefinition) // ThenInclude - jeszcze dołączamy definicje cech

.Where(i => i.Inventory != null) //zabezpieczenie

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrument == null)

{

return NotFound();

}

return View(instrument);

}

// GET: InstrumentInventories/Create

public IActionResult Create()

{

ViewBag.SupplierList = new SelectList(\_context.Supplier, "SupplierId", "Name");

return View();

}

// POST: InstrumentInventories/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost, ValidateAntiForgeryToken]

public async Task<IActionResult> Create(

int? supplierId,

string action,

List<int> selectedInstrumentIds)

{

ViewBag.SupplierList = new SelectList(\_context.Supplier, "SupplierId", "Name", supplierId);

if (action == "ShowSupplierInstruments")

{

if (supplierId == null)

{

ModelState.AddModelError("supplierId", "Musisz wybrać dostawcę.");

}

else

{

ViewBag.Instruments = \_context.Instrument

.Where(i => i.SupplierId == supplierId)

.ToList();

// PRZEKAZUJEMY supplierId osobno:

ViewBag.SelectedSupplierId = supplierId;

}

return View();

}

// „Dalej”

if (selectedInstrumentIds == null || !selectedInstrumentIds.Any())

{

ModelState.AddModelError("", "Musisz zaznaczyć przynajmniej jeden instrument.");

ViewBag.Instruments = \_context.Instrument

.Where(i => i.SupplierId == supplierId)

.ToList();

ViewBag.SelectedSupplierId = supplierId;

return View();

}

//przekieruj do Documents/Create z parametrami

return RedirectToAction(

actionName: "Create",

controllerName: "Documents",

routeValues: new

{

supplierId = supplierId,

selectedInstrumentIds = selectedInstrumentIds

}

);

}

// GET: InstrumentInventories/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var instrument = await \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Supplier)

.Include(i => i.Inventory)

.Include(i => i.InstrumentFeatures)

.ThenInclude(ifeat => ifeat.FeatureDefinition) // ThenInclude - jeszcze dołączamy definicje cech

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrument == null)

{

return NotFound();

}

ViewBag.SupplierList = new SelectList(

\_context.Supplier, "SupplierId", "Name");

ViewBag.CategoryList = new SelectList(

\_context.Category, "CategoryId", "Name");

return View(instrument);

}

// POST: Instruments/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("InstrumentId,Name,Price,Description,EAN,SKU,SerialNumber,Inventory,SupplierId,CategoryId,InstrumentFeatures")] Instrument instrument)

{

if (id != instrument.InstrumentId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(instrument);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!InstrumentExists(instrument.InstrumentId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["CategoryId"] = new SelectList(\_context.Category, "CategoryId", "CategoryId", instrument.CategoryId);

ViewData["SupplierId"] = new SelectList(\_context.Supplier, "SupplierId", "SupplierId", instrument.SupplierId);

return View(instrument);

}

// GET: InstrumentInventories/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var instrumentInventory = await \_context.InstrumentInventory

.Include(i => i.Instrument)

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrumentInventory == null)

{

return NotFound();

}

return View(instrumentInventory);

}

// POST: InstrumentInventories/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var instrumentInventory = await \_context.InstrumentInventory

.Include(i => i.Instrument)

.FirstOrDefaultAsync(m => m.InstrumentId == id); ;

if (instrumentInventory != null)

{

\_context.InstrumentInventory.Remove(instrumentInventory);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool InstrumentExists(int id)

{

return \_context.InstrumentInventory.Any(e => e.InstrumentInventoryId == id);

}

}

}

## InstrumentsController – crud dla istrumentów w ogólnej sieci dystrybucji

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.Data.SqlClient;

using Microsoft.EntityFrameworkCore;

using Music\_Store\_Warehouse\_App.Data;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers

{

public class InstrumentsController : Controller

{

private readonly Music\_Store\_Warehouse\_AppContext \_context;

public InstrumentsController(Music\_Store\_Warehouse\_AppContext context)

{

\_context = context;

}

// GET: Instruments

public async Task<IActionResult> Index(

string sortOrder,

string currentFilter,

string searchString,

int? pageNumber,

int? categoryId,

int? supplierId)

{

ViewData["CurrentSort"] = sortOrder;

ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ? "name\_desc" : "";

ViewData["PriceSortParm"] = sortOrder == "Price" ? "price\_desc" : "Price";

ViewData["CurrentFilter"] = searchString;

if (searchString != null)

{

pageNumber = 1;

}

else

{

searchString = currentFilter;

}

var categories = await \_context.Category

.OrderBy(c => c.Name)

.ToListAsync();

var suppliers = await \_context.Supplier

.OrderBy(s => s.Name)

.ToListAsync();

ViewData["CategoryList"] = new SelectList(categories, "CategoryId", "Name", categoryId);

ViewData["CurrentCategory"] = categoryId; // by wiedzieć, która opcja ma być selected

ViewData["SupplierList"] = new SelectList(suppliers, "SupplierId", "Name", supplierId);

ViewData["CurrentSupplier"] = supplierId; // by wiedzieć, która opcja ma być selected

IQueryable<Instrument> instruments = \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Inventory)

.Include(i => i.Supplier);

// --- Filtrowanie po wyszukiwanej frazie ---

if (!String.IsNullOrEmpty(searchString))

{

instruments = instruments.Where(s =>

s.Name.Contains(searchString) ||

s.Supplier.Name.Contains(searchString));

}

// --- Filtrowanie po kategorii, jeśli użytkownik wybrał categoryId ---

if (categoryId.HasValue)

{

instruments = instruments.Where(i => i.CategoryId == categoryId.Value);

}

if (supplierId.HasValue)

{

instruments = instruments.Where(i => i.SupplierId == supplierId.Value);

}

switch (sortOrder)

{

case "name\_desc":

instruments = instruments.OrderByDescending(s => s.Name);

break;

case "Price":

instruments = instruments.OrderBy(s => s.Price);

break;

case "price\_desc":

instruments = instruments.OrderByDescending(s => s.Price);

break;

default:

instruments = instruments.OrderBy(s => s.Name);

break;

}

int pageSize = 10;

return View(await PaginatedList<Instrument>.CreateAsync(instruments.AsNoTracking(), pageNumber ?? 1, pageSize));

}

// GET: Instruments/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var instrument = await \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Supplier)

.Include( i => i.Inventory)

.Include(i => i.InstrumentFeatures)

.ThenInclude(ifeat => ifeat.FeatureDefinition) // ThenInclude - jeszcze dołączamy definicje cech

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrument == null)

{

return NotFound();

}

return View(instrument);

}

// GET: Instruments/Create

[HttpGet]

public IActionResult Create()

{

PrepareViewBags();

return View(new Instrument());

}

// POST: Instruments/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(

[Bind("InstrumentId,Name,Price,Description,EAN,SKU,SerialNumber,Quantity,SupplierId,CategoryId")] Instrument instrument, //ważne BIND - aby nie przekazywać za dużo danych

string action, // "ShowFeatures" lub "SaveInstrument"

[FromForm] IList<InstrumentFeature> InstrumentFeatures)

{

// Zawsze przygotowujemy ViewBag dropdownów, bo widok ich potrzebuje

PrepareViewBags();

// Jeżeli użytkownik wcisnął „Pokaż cechy”

if (action == "ShowFeatures")

{

// Upewnijmy się przynajmniej, że wybrano kategorię

if (instrument.CategoryId == 0)

{

ModelState.AddModelError(nameof(instrument.CategoryId), "Kategoria jest wymagana, aby wyświetlić cechy.");

}

else

{

// Jeżeli jest wartość CategoryId, ładujemy od razu cechy – bez sprawdzania pozostałych błędów

var cat = \_context.Category.Find(instrument.CategoryId);

if (cat != null)

{

var typeEnum = Enum.Parse<FType>(cat.Name, ignoreCase: true);

ViewBag.FeatureDefinitions = \_context.FeatureDefinition

.Where(f => f.Type == typeEnum)

.ToList();

}

}

// Zwracamy widok z aktualnym modelem, nawet jeśli np. Name lub Price są puste.

// Gdy wracamy tym View(instrument), wszystkie wpisane dotąd pola w formularzu będą w polach <input asp-for="…"/>,

// a ViewBag.FeatureDefinitions wyświetli tabelę cech (o ile CategoryId != 0).

return View(instrument);

}

// Jeżeli użytkownik wcisnął „Zapisz instrument”

else if (action == "SaveInstrument")

{

// Przypisz z POST-a listę cech (może być pusta, jeżeli nie było tabeli)

instrument.InstrumentFeatures = InstrumentFeatures;

// Teraz wykonujemy normalną walidację całego modelu

if (!ModelState.IsValid)

{

// Aby tabela cech nie zniknęła, musimy znów załadować FeatureDefinitions

if (instrument.CategoryId != 0)

{

var cat = \_context.Category.Find(instrument.CategoryId);

var typeEnum = Enum.Parse<FType>(cat.Name, ignoreCase: true);

ViewBag.FeatureDefinitions = \_context.FeatureDefinition

.Where(f => f.Type == typeEnum)

.ToList();

}

return View(instrument);

}

// Jeśli ModelState jest OK – zapisujemy do bazy:

\_context.Add(instrument);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

// Jeśli action nie został przekazany (lub ma nietypową wartość), po prostu wyświetlamy formularz

return View(instrument);

}

private void PrepareViewBags()

{

ViewBag.SupplierList = new SelectList(

\_context.Supplier, "SupplierId", "Name");

ViewBag.CategoryList = new SelectList(

\_context.Category, "CategoryId", "Name");

}

// GET: Instruments/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var instrument = await \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Supplier)

.Include(i => i.InstrumentFeatures)

.ThenInclude(ifeat => ifeat.FeatureDefinition) // ThenInclude - jeszcze dołączamy definicje cech

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrument == null)

{

return NotFound();

}

PrepareViewBags();

return View(instrument);

}

// POST: Instruments/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("InstrumentId,Name,Price,Description,EAN,SKU,SerialNumber,Quantity,SupplierId,CategoryId,InstrumentFeatures")] Instrument instrument)

{

if (id != instrument.InstrumentId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(instrument);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!InstrumentExists(instrument.InstrumentId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["CategoryId"] = new SelectList(\_context.Category, "CategoryId", "CategoryId", instrument.CategoryId);

ViewData["SupplierId"] = new SelectList(\_context.Supplier, "SupplierId", "SupplierId", instrument.SupplierId);

return View(instrument);

}

// GET: Instruments/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var instrument = await \_context.Instrument

.Include(i => i.Category)

.Include(i => i.Supplier)

.Include(i => i.InstrumentFeatures)

.FirstOrDefaultAsync(m => m.InstrumentId == id);

if (instrument == null)

{

return NotFound();

}

return View(instrument);

}

// POST: Instruments/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var instrument = await \_context.Instrument.FindAsync(id); //instrument features też się usunie poprzez delete Cascade w DB (konfiguracja w pliku db context)

if (instrument != null)

{

\_context.Instrument.Remove(instrument);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool InstrumentExists(int id)

{

return \_context.Instrument.Any(e => e.InstrumentId == id);

}

}

}

## SuppliersController – zarządzanie dostawcami instrumentów

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Music\_Store\_Warehouse\_App.Data;

using Music\_Store\_Warehouse\_App.Models;

namespace Music\_Store\_Warehouse\_App.Controllers

{

public class SuppliersController : Controller

{

private readonly Music\_Store\_Warehouse\_AppContext \_context;

public SuppliersController(Music\_Store\_Warehouse\_AppContext context)

{

\_context = context;

}

// GET: Suppliers

public async Task<IActionResult> Index()

{

var music\_Store\_Warehouse\_AppContext = \_context.Supplier.Include(s => s.Address);

return View(await music\_Store\_Warehouse\_AppContext.ToListAsync());

}

// GET: Suppliers/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var supplier = await \_context.Supplier

.Include(s => s.Address)

.FirstOrDefaultAsync(m => m.SupplierId == id);

if (supplier == null)

{

return NotFound();

}

return View(supplier);

}

// GET: Suppliers/Create

public IActionResult Create()

{

return View();

}

// POST: Suppliers/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Supplier supplier)

{

if (ModelState.IsValid)

{

\_context.Add(supplier);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(supplier);

}

// GET: Suppliers/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var supplier = await \_context.Supplier

.Include(a => a.Address) //Dolejamy adres, ponieważ ten edit ma modyfikować i dostawcę i adres

.FirstOrDefaultAsync(s => s.SupplierId == id);

if (supplier == null)

{

return NotFound();

}

return View(supplier);

}

// POST: Suppliers/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, Supplier supplier) //Brak bindowania ponieważ nie działało przesłanie Address i jego pól

{

if (id != supplier.SupplierId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

// Pobierz z bazy istniejący Supplier wraz z powiązanym Address

var existing = await \_context.Supplier

.Include(s => s.Address)

.FirstOrDefaultAsync(s => s.SupplierId == id);

if (existing == null)

return NotFound();

// Zaktualizuj pola dostawcy

existing.Name = supplier.Name;

existing.Email = supplier.Email;

// Jeżeli Address bywa null, utwórz nowy:

if (existing.Address == null)

{

existing.Address = new Address();

existing.Address.SupplierId = existing.SupplierId;

}

// Zaktualizuj pola istniejącego Address

existing.Address.Street = supplier.Address.Street;

existing.Address.City = supplier.Address.City;

existing.Address.PostalCode = supplier.Address.PostalCode;

// Teraz EF Core zrozumie, że:

// - dostawca istnieje → zmienia tylko jego kolumny

// - Address istnieje (ma ustawione AddressId i SupplierId) → zmienia tylko kolumny Street/City/PostalCode

// Nie będzie próbował wstawić nowego wiersza do tabeli Address.

\_context.Update(existing);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateConcurrencyException)

{

if (!SupplierExists(supplier.SupplierId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(supplier);

}

// GET: Suppliers/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var supplier = await \_context.Supplier

.Include(s => s.Address)

.FirstOrDefaultAsync(m => m.SupplierId == id);

if (supplier == null)

{

return NotFound();

}

return View(supplier);

}

// POST: Suppliers/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var supplier = await \_context.Supplier.FindAsync(id);

if (supplier != null)

{

\_context.Supplier.Remove(supplier);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool SupplierExists(int id)

{

return \_context.Supplier.Any(e => e.SupplierId == id);

}

}

}

# Widoki

## Documents

### Create – utworzenie dokumentu na podstawie dostawcy i listy instrumentów od tego dostawcy

@model Music\_Store\_Warehouse\_App.Models.Document

@{

ViewData["Title"] = "Create Document";

}

<div class="container my-4">

<h1 class="mb-4">Nowy dokument PZ</h1>

<h3>Dostawca: @ViewBag.supplier.Name</h3>

<h4>Adres Dostawcy: </h4>

<h5>

<ol>Kod pocztowy - @ViewBag.supplier.Address.PostalCode</ol>

<ol>Miasto - @ViewBag.supplier.Address.City</ol>

<ol>Ulica - @ViewBag.supplier.Address.Street</ol>

</h5>

<**form** **asp-action**="Create" method="post">

@if (Model.DocumentInstruments != null && Model.DocumentInstruments.Any())

{

var items = Model.DocumentInstruments.ToList();

<h4 class="mt-4 mb-3">Instrumenty w dokumencie</h4>

<div class="table-responsive">

<table class="table table-bordered table-hover">

<thead class="thead-light">

<tr>

<th>Nazwa instrumentu</th>

<th>Kategoria</th>

<th>Dostawca</th>

<th style="width: 8rem;">Ilość</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < items.Count; i++)

{

<tr>

<td>

@items[i].Instrument.Name

<input type="hidden"

name="DocumentInstruments[@i].InstrumentId"

value="@items[i].InstrumentId" />

</td>

<td>@items[i].Instrument.Category.Name</td>

<td>@items[i].Instrument.Supplier?.Name</td>

<td>

<input type="number"

name="DocumentInstruments[@i].Quantity"

value="@items[i].Quantity"

class="form-control"

min="1" />

</td>

</tr>

}

</tbody>

</table>

</div>

}

else

{

<p>Brak instrumentów przekazanych do dokumentu</p>

}

<div class="mt-4">

<button type="submit" class="btn btn-primary">Zapisz dokument</button>

<**a** **asp-action**="Index" class="btn btn-secondary ms-2">Anuluj</**a**>

</div>

</**form**>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial");

}

### Delete – usunięcie dokumentu

@model Music\_Store\_Warehouse\_App.Models.Document

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Document</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Type)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Type)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Date)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Date)

</dd>

</dl>

<**form** **asp-action**="Delete">

<**input** **type**="hidden" **asp-for**="DocumentId" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index">Back to List</**a**>

</**form**>

</div>

### Details – szczegóły dokumentu, wraz z danymi o instrumentach które są jako pozycje dokumentu i podsumowaniami ilości i wartości pozycji na dokumencie

@model Music\_Store\_Warehouse\_App.Models.Document

@{

ViewData["Title"] = "Details";

var supplier = (Supplier)ViewBag.Supplier;

}

<h1>Details</h1>

<div>

<h4>Dokument</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Type)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Type)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Date)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Date)

</dd>

@if(supplier != null){

<dt class="col-sm-2">

Nazwa

</dt>

<dd class="col-sm-10">

@ViewBag.Supplier.Name

</dd>

<dt class="col-sm-2">

Adres Dostawcy

</dt>

<dd class="col-sm-10">

Kod pocztowy - @ViewBag.Supplier.Address.PostalCode

Miasto - @ViewBag.Supplier.Address.City

Ulica - @ViewBag.Supplier.Address.Street

</dd>

<dt class="col-sm-2">

Email

</dt>

<dd class="col-sm-10">

@ViewBag.Supplier.Email)

</dd>

}

<dt class="col-sm-2">

Wartość dokumentu

</dt>

<dd class="col-sm-10">

@Model.DocumentInstruments.Sum(di => di.Instrument.Price \* di.Quantity)

</dd>

</dl>

@if (Model.DocumentInstruments != null && Model.DocumentInstruments.Any())

{

var items = Model.DocumentInstruments.ToList();

<h4 class="mt-4 mb-3">Instrumenty w dokumencie</h4>

<div class="table-responsive">

<table class="table table-bordered table-hover">

<thead class="thead-light">

<tr>

<th>Nazwa instrumentu</th>

<th>Kategoria</th>

<th>Cena</th>

<th style="width: 8rem;">Ilość</th>

<th style="width: 8rem;">Suma</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < items.Count; i++)

{

<tr>

<td>

@items[i].Instrument.Name

<input type="hidden"

name="DocumentInstruments[@i].InstrumentId"

value="@items[i].InstrumentId" />

</td>

<td>

@items[i].Instrument.Category.Name

</td>

<td>

@items[i].Instrument.Price

</td>

<td>

@items[i].Quantity

</td>

<td>

@(items[i].Quantity \* @items[i].Instrument.Price)

</td>

</tr>

}

</tbody>

</table>

</div>

}

else

{

<p>Brak instrumentów przekazanych do dokumentu</p>

}

</div>

<div>

<**a** **asp-action**="Edit" **asp-route-id**="@Model?.DocumentId">Edit</**a**> |

<**a** **asp-action**="Index">Back to List</**a**>

</div>

### Edit – edycja dokumentu

@model Music\_Store\_Warehouse\_App.Models.Document

@{

ViewData["Title"] = "Edit";

}

<h1>Edit</h1>

<h4>Operator nie ma dostępu do korekty dokumentów</h4>

@\*

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="DocumentId" />

<div class="form-group">

<label asp-for="Type" class="control-label"></label>

<select asp-for="Type" class="form-control"></select>

<span asp-validation-for="Type" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Date" class="control-label"></label>

<input asp-for="Date" class="form-control" />

<span asp-validation-for="Date" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</form>

</div>

</div>

\*@

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Index – wyświetlenie wszystkich dokumentów

@model IEnumerable<Music\_Store\_Warehouse\_App.Models.Document>

@{

ViewData["Title"] = "Index";

}

<h1>Dokumenty</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Type)

</th>

<th>

@Html.DisplayNameFor(model => model.Date)

</th>

<th>

@Html.DisplayName("Ilość artykułów")

</th>

<th>

@Html.DisplayName("Wartość artykułów")

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Type)

</td>

<td>

@Html.DisplayFor(modelItem => item.Date)

</td>

<td>

@(item.DocumentInstruments?.Sum(i => i.Quantity) ?? 0)

</td>

<td>

@(item.DocumentInstruments?.Sum(i => i.Instrument.Price \* i.Quantity) ?? 0)

</td>

<td>

<**a** **asp-action**="Edit" **asp-route-id**="@item.DocumentId">Edit</**a**> |

<**a** **asp-action**="Details" **asp-route-id**="@item.DocumentId">Details</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.DocumentId">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

## Home

### About – opis aplikacji

@{

ViewData["Title"] = "O Aplikacji";

}

<h1>@ViewData["Title"]</h1>

<p>

Aplikacja to internetowy system zarządzania magazynem sklepu muzycznego. Został on stworzony na potrzeby zaliczenia przedmiotu

"Programowanie Zaawansowane" na VI semestrze studiów na kierunku informatyka w Siedlcach.

Umożliwia on dodawanie, edytowanie i usuwanie instrumentów, kategorii oraz dostawców, a także definiowanie i modyfikowanie

szczegółowych cech każdego instrumentu. Pozwala na przeszukiwanie, sortowanie i stronicowanie listy produktów,

a w widokach szczegółowych pokazuje powiązane informacje

</p>

### Privacy – strona dotycząca prywatność na stronie

@{

ViewData["Title"] = "Polityka prywatności";

}

<h1>@ViewData["Title"]</h1>

<p>Dane użytkowników nie są przetwarzane na tej stronie.</p>

### Index – widok strony głównej

@{

ViewData["Title"] = "Strona główna";

}

<div class="jumbotron">

<h1 class="mb-5">Zarządzanie magazynem muzycznym</h1>

</div>

<div class="row">

<div class="col-md-4">

<h5>Witaj w aplikacji służącej do zarządzania asortymentem muzycznym</h5>

<p>

To prosta aplikacja umożliwiająca dodawanie i zarządzanie artykułami oraz dostawcami z branży muzycznej.

</p>

</div>

<a href="/Instruments"

class="col-md-4 d-block text-dark text-decoration-none">

<div class="kafelek">

<h2>Instrumenty</h2>

<p>Przeglądaj oferty instrumentów na rynku.</p>

</div>

</a>

<a href="/Suppliers"

class="col-md-4 d-block text-dark text-decoration-none">

<div class="kafelek">

<h2>Dostawcy</h2>

<p>Zarządzaj bazą dostawców.</p>

</div>

</a>

</div>

<div class="row">

<div class="col-md-4"></div> <!-- spacer -->

<a href="/InstrumentInventories"

class="col-md-4 d-block text-dark text-decoration-none">

<div class="kafelek">

<h2>Magazyn</h2>

<p>Dodawaj, wydawaj oraz edytuj instrumenty przechowywane w magazynie.</p>

</div>

</a>

<a href="/Documents"

class="col-md-4 d-block text-dark text-decoration-none">

<div class="kafelek">

<h2>Dokumenty</h2>

<p>Dokumenty zarządzające stanem magazynowym, dodawaj asortyment na podstawie dokumentów PZ.</p>

</div>

</a>

</div>

## InstrumentFeatures

### Create – utworzenie cechy dla danego instrumentu na podstawie definicji cechy.

@model Music\_Store\_Warehouse\_App.Models.InstrumentFeature

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>InstrumentFeature</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Create">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="InstrumentId" class="control-label"></**label**>

<**select** **asp-for**="InstrumentId" class ="form-control" **asp-items**="ViewBag.InstrumentId"></**select**>

</div>

<div class="form-group">

<**label** **asp-for**="FeatureDefinitionId" class="control-label"></**label**>

<**select** **asp-for**="FeatureDefinitionId" class ="form-control" **asp-items**="ViewBag.FeatureDefinitionId"></**select**>

</div>

<div class="form-group">

<**label** **asp-for**="Value" class="control-label"></**label**>

<**input** **asp-for**="Value" class="form-control" />

<**span** **asp-validation-for**="Value" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Delete – usunięcie cechy danego instrumentu

@model Music\_Store\_Warehouse\_App.Models.InstrumentFeature

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>InstrumentFeature</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Instrument)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Instrument.InstrumentId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.FeatureDefinition)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.FeatureDefinition.FeatureDefinitionId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Value)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Value)

</dd>

</dl>

<**form** **asp-action**="Delete">

<**input** **type**="hidden" **asp-for**="InstrumentFeatureId" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index">Back to List</**a**>

</**form**>

</div>

### Details – szczegóły cechy instrumentu

@model Music\_Store\_Warehouse\_App.Models.InstrumentFeature

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>InstrumentFeature</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Instrument)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Instrument.InstrumentId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.FeatureDefinition)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.FeatureDefinition.FeatureDefinitionId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Value)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Value)

</dd>

</dl>

</div>

<div>

<**a** **asp-action**="Edit" **asp-route-id**="@Model?.InstrumentFeatureId">Edit</**a**> |

<**a** **asp-action**="Index">Back to List</**a**>

</div>

### Edit – edycja cechy instrumentu

@model Music\_Store\_Warehouse\_App.Models.InstrumentFeature

@{

ViewData["Title"] = "Edit";

}

<h1>Edit</h1>

<h4>InstrumentFeature</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Edit">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<**input** **type**="hidden" **asp-for**="InstrumentFeatureId" />

<div class="form-group">

<**label** **asp-for**="InstrumentId" class="control-label"></**label**>

<**select** **asp-for**="InstrumentId" class="form-control" **asp-items**="ViewBag.InstrumentId"></**select**>

<**span** **asp-validation-for**="InstrumentId" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="FeatureDefinitionId" class="control-label"></**label**>

<**select** **asp-for**="FeatureDefinitionId" class="form-control" **asp-items**="ViewBag.FeatureDefinitionId"></**select**>

<**span** **asp-validation-for**="FeatureDefinitionId" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Value" class="control-label"></**label**>

<**input** **asp-for**="Value" class="form-control" />

<**span** **asp-validation-for**="Value" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Index – widok cech instrumentów

@model IEnumerable<Music\_Store\_Warehouse\_App.Models.InstrumentFeature>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Instrument)

</th>

<th>

@Html.DisplayNameFor(model => model.FeatureDefinition)

</th>

<th>

@Html.DisplayNameFor(model => model.Value)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Instrument.InstrumentId)

</td>

<td>

@Html.DisplayFor(modelItem => item.FeatureDefinition.FeatureDefinitionId)

</td>

<td>

@Html.DisplayFor(modelItem => item.Value)

</td>

<td>

<**a** **asp-action**="Edit" **asp-route-id**="@item.InstrumentFeatureId">Edit</**a**> |

<**a** **asp-action**="Details" **asp-route-id**="@item.InstrumentFeatureId">Details</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.InstrumentFeatureId">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

## InstrumentInventories – główna obsługa magazynu czyli najważniejszy widok w projekcie. służy do dodawania instrumentów na podstawie dokumentów przyjęcia zewnętrznego. można w nim zarządzać instrumentami i jego stanami magazynowymi.

### Create – utworzenie nowego przyjęcia magazynowego na podstawie instrumentów od danego dostawcy przy pomocy dokumentu przyjęcia magazynowego

@model object

@using Music\_Store\_Warehouse\_App.Models

@{

ViewData["Title"] = "Create";

var instruments = ViewBag.Instruments as List<Instrument>;

var selSupplier = (int?)ViewBag.SelectedSupplierId;

}

<div class="container my-4">

<h1 class="mb-4">Przyjęcie zewnętrzne</h1>

<**form** method="post" **asp-action**="Create">

<div class="row">

<div class="col-md-6 col-lg-4">

<div class="form-group">

<label for="supplierId">Dostawca</label>

<**select** **name**="supplierId"

class="form-control"

**asp-items**="ViewBag.SupplierList">

<**option** **value**="">— (brak) —</**option**>

</**select**>

</div>

<div class="mt-3">

<button type="submit"

name="action"

value="ShowSupplierInstruments"

class="btn btn-secondary btn-block mb-2">

Pokaż instrumenty

</button>

@if (instruments != null && instruments.Any())

{

<input type="hidden" name="supplierId" value="@selSupplier" />

<button type="submit"

name="action"

value="Dalej"

class="btn btn-primary btn-block">

Dalej

</button>

}

</div>

</div>

</div>

@if (instruments != null && instruments.Any())

{

<div class="mt-5">

<h4 class="mb-3">Instrumenty dostawcy</h4>

<div class="table-responsive">

<table class="table table-hover">

<thead class="thead-light">

<tr>

<th style="width: 1rem;"></th>

<th>Nazwa</th>

<th style="width: 8rem;" class="text-right">Cena</th>

</tr>

</thead>

<tbody>

@foreach (var inst in instruments)

{

<tr>

<td>

<input type="checkbox"

name="selectedInstrumentIds"

value="@inst.InstrumentId" />

</td>

<td>@inst.Name</td>

<td class="text-right">@inst.Price.ToString("C")</td>

</tr>

}

</tbody>

</table>

</div>

</div>

}

</**form**>

<div class="mt-4">

<**a** **asp-action**="Index" class="btn btn-link">&larr; Powrót do listy</**a**>

</div>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

### Delete – usnięcie instrumentu z magazynu

@model Music\_Store\_Warehouse\_App.Models.InstrumentInventory

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>InstrumentInventory</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Instrument)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Instrument.Name)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Quantity)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Quantity)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.LastUpdated)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.LastUpdated)

</dd>

</dl>

<**form** **asp-action**="Delete">

<**input** **type**="hidden" **asp-for**="InstrumentInventoryId" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index">Back to List</**a**>

</**form**>

</div>

### Details – szczegóły instrumentu z magazynu

@model Music\_Store\_Warehouse\_App.Models.Instrument

@{

ViewData["Title"] = "Szczegóły";

}

<h1>Szczegóły</h1>

<h4>Instrument</h4>

<hr />

<div class="container">

<div class="row">

<div class="col-md-6">

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Price)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Price)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Description)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Description)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.EAN)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.EAN)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.SKU)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.SKU)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Inventory.Quantity)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Inventory.Quantity)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Supplier)

</dt>

<dd class="col-sm-10">

@if (Model.Supplier != null)

{

<!-- Przekierowanie do szczegółów dostawcy za pomocą route-id-->

<a asp-controller="Suppliers" asp-action="Details" asp-route-id="@Model.Supplier.SupplierId">

@Model.Supplier.Name

</a>

}

else

{

<text>Brak przypisanego dostawcy</text>

}

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Category)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Category.Name)

</dd>

</dl>

</div>

<!-- Prawa kolumna: tabela z cechami instrumentu -->

<div class="col-md-6">

<div class="card mb-4">

<div class="card-header bg-light">

<strong>Cechy szczegółowe</strong>

</div>

<div class="card-body p-0">

@if (Model.InstrumentFeatures != null && Model.InstrumentFeatures.Any())

{

<table class="table table-striped mb-0">

<thead class="thead-light">

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@foreach (var feat in Model.InstrumentFeatures)

{

<tr>

<td>

@feat.FeatureDefinition?.Name

</td>

<td>

@feat.Value

</td>

</tr>

}

</tbody>

</table>

}

else

{

<div class="p-3">

<em>Brak przypisanych cech dla tego instrumentu.</em>

</div>

}

</div>

</div>

</div>

</div>

</div>

<div>

<**a** **asp-action**="Edit" **asp-route-id**="@Model?.InstrumentId" class="btn btn-sm btn-primary">Edit</**a**> |

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

### Edit – edycja instrumentów z magazynu

@model Music\_Store\_Warehouse\_App.Models.Instrument

@{

ViewData["Title"] = "Edycja";

}

<h1>Edycja</h1>

<h4>Instrument</h4>

<hr />

<**form** **asp-action**="Edit">

<div class="row">

<div class="col-md-4">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<**input** **type**="hidden" **asp-for**="InstrumentId" />

<div class="form-group">

<**label** **asp-for**="Name" class="control-label"></**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Price" class="control-label"></**label**>

<**input** **asp-for**="Price" class="form-control" **type**="number" />

<**span** **asp-validation-for**="Price" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Description" class="control-label"></**label**>

<**input** **asp-for**="Description" class="form-control" />

<**span** **asp-validation-for**="Description" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="EAN" class="control-label"></**label**>

<**input** **asp-for**="EAN" class="form-control" />

<**span** **asp-validation-for**="EAN" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="SKU" class="control-label"></**label**>

<**input** **asp-for**="SKU" class="form-control" />

<**span** **asp-validation-for**="SKU" class="text-danger"></**span**>

</div>

<div class="form-group">

<**input** **type**="hidden" **asp-for**="Inventory.InstrumentInventoryId" />

<**input** **type**="hidden" **asp-for**="Inventory.InstrumentId" />

<**label** **asp-for**="Inventory.Quantity" class="control-label"></**label**>

<**input** **asp-for**="Inventory.Quantity" class="form-control" />

<**span** **asp-validation-for**="Inventory.Quantity" class="text-danger"></**span**>

</div>

<div class="form-group">

<label asp-for="SupplierId">Dostawca</label>

<select asp-for="SupplierId"

asp-items="ViewBag.SupplierList"

class="form-control">

<option value="">— (brak) —</option>

</select>

</div>

<div class="form-group">

<label asp-for="CategoryId">Kategoria</label>

<select asp-for="CategoryId"

asp-items="ViewBag.CategoryList"

class="form-control">

<option value="">— wybierz —</option>

</select>

<span asp-validation-for="CategoryId" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary mt-3 mb-2" />

</div>

</div>

<div class="col-md-2">

<!-- rozdzielacz -->

</div>

<div class="col-md-6">

<!-- Tabela z cechami instrumentu -->

@if (Model.InstrumentFeatures != null && Model.InstrumentFeatures.Any())

{

var featuresList = Model.InstrumentFeatures.ToList();

<h4>Przypisane cechy</h4>

<table class="table table-bordered">

<thead>

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < featuresList.Count; i++)

{

// Pojedyncza cecha z listy

var ifeat = featuresList[i];

<tr>

<td>

@ifeat.FeatureDefinition.Name

<!-- Ukryte pola, żeby ASP.NET wiedział, który InstrumentFeature edytujemy -->

<input type="hidden"

name="InstrumentFeatures[@i].InstrumentFeatureId"

value="@ifeat.InstrumentFeatureId" />

<input type="hidden"

name="InstrumentFeatures[@i].FeatureDefinitionId"

value="@ifeat.FeatureDefinitionId" />

</td>

<td>

<!-- Ręcznie składamy pole z name="InstrumentFeatures[i].Value" -->

<input type="text"

name="InstrumentFeatures[@i].Value"

value="@ifeat.Value"

class="form-control" />

</td>

</tr>

}

</tbody>

</table>

}

else

{

<p>Ten instrument nie ma jeszcze przypisanych cech.</p>

}

</div>

</div>

</**form**>

<div>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

### Index – widok instrumentów które są w magazynie

@using Music\_Store\_Warehouse\_App.Data

@model PaginatedList<Music\_Store\_Warehouse\_App.Models.Instrument>

@{

ViewData["Title"] = "Magazyn";

}

<h1>Magazyn</h1>

<p>

<**a** **asp-action**="Create" class="btn btn-sm btn-success">Przyjęcie Zewnętrzne</**a**>

</p>

<**form** **asp-action**="Index" method="get" class="form-inline mb-3">

<p class="mb-0">

<!-- Pole wyszukiwania -->

<label class="mr-2">

Szukana nazwa:

<input type="text"

name="searchString"

value="@(ViewData["CurrentFilter"] ?? "")"

class="form-control form-control-sm ml-1"

style="width: 200px;" />

</label>

<!-- Dropdown kategorii -->

<label class="ml-3 mr-2">

kategoria:

<select name="categoryId"

id="categoryId"

class="form-control form-control-sm ml-1"

style="width: 180px;">

<**option** **value**="">Wszystkie</**option**>

@foreach (SelectListItem cat in (SelectList)ViewData["CategoryList"])

{

<**option** **value**="@cat.Value" selected="@(cat.Selected ? "selected" : null)">

@cat.Text

</**option**>

}

</select>

</label>

<!-- Dropdown dostawców -->

<label class="ml-3 mr-2">

dostawca:

<select name="supplierId"

id="supplierId"

class="form-control form-control-sm ml-1"

style="width: 180px;">

<**option** **value**="">Wszyscy</**option**>

@foreach (SelectListItem sup in (SelectList)ViewData["SupplierList"])

{

<option value="@sup.Value" selected="@(sup.Selected ? "selected" : null)">

@sup.Text

</option>

}

</select>

</label>

<!-- Przycisk wyszukiwania i link do pełnej listy -->

<button type="submit" class="btn btn-default">Szukaj</button> |

<a asp-action="Index">Wyczyść</a>

</p>

</form>

<table class="table">

<thead>

<tr>

<th>

<a asp-action="Index" asp-route-sortOrder="@ViewData["NameSortParm"]" asp-route-currentFilter="@ViewData["CurrentFilter"]">Name</a>

</th>

<th>

<a asp-action="Index" asp-route-sortOrder="@ViewData["PriceSortParm"]" asp-route-currentFilter="@ViewData["CurrentFilter"]">Price</a>

</th>

<th>

<!-- Dodajemy [0] ponieaż teraz model to PaginatedList która nie ma właściwości Description tak jak wcześniejszy

model instrument.

\*/ -->

@Html.DisplayNameFor(model => model[0].Description)

</th>

<th>

@Html.DisplayNameFor(model => model[0].EAN)

</th>

<th>

@Html.DisplayNameFor(model => model[0].SKU)

</th>

<th>

@Html.DisplayNameFor(model => model[0].Inventory.Quantity)

</th>

<th>

@Html.DisplayNameFor(model => model[0].Supplier)

</th>

<th>

@Html.DisplayNameFor(model => model[0].Category)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

@Html.DisplayFor(modelItem => item.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.EAN)

</td>

<td>

@Html.DisplayFor(modelItem => item.SKU)

</td>

<td>

@Html.DisplayFor(modelItem => item.Inventory.Quantity)

</td>

<td>

@Html.DisplayFor(modelItem => item.Supplier.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Category.Name)

</td>

<td style>

<**a** **asp-action**="Edit" **asp-route-id**="@item.InstrumentId" class="btn btn-sm btn-primary">Edytuj</**a**> |

<**a** **asp-action**="Details" **asp-route-id**="@item.InstrumentId" class="btn btn-sm btn-info">Szczegóły</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.InstrumentId" class="btn btn-sm btn-danger">Usuń</**a**>

</td>

</tr>

}

</tbody>

</table>

@{

var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";

var nextDisabled = !Model.HasNextPage ? "disabled" : "";

}

<**a** **asp-action**="Index"

**asp-route-sortOrder**="@ViewData["CurrentSort"]"

**asp-route-pageNumber**="@(Model.PageIndex - 1)"

**asp-route-currentFilter**="@ViewData["CurrentFilter"]"

class="btn btn-default @prevDisabled">

Previous

</**a**>

<**a** **asp-action**="Index"

**asp-route-sortOrder**="@ViewData["CurrentSort"]"

**asp-route-pageNumber**="@(Model.PageIndex + 1)"

**asp-route-currentFilter**="@ViewData["CurrentFilter"]"

class="btn btn-default @nextDisabled">

Next

</**a**>

## Instruments

### Create – tworzenie isntrumentu w systemie

@model Music\_Store\_Warehouse\_App.Models.Instrument

@using Music\_Store\_Warehouse\_App.Models

<h1>Dodaj nowy instrument</h1>

<**form** method="post" **asp-action**="Create">

<div class="container">

<div class="row">

<div class="col-md-6">

@\* Główne pola Instrument: \*@

<div class="form-row">

<div class="form-group col-md-6">

<**label** **asp-for**="Name"></**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group col-md-6">

<**label** **asp-for**="Price"></**label**>

<**input** **asp-for**="Price" class="form-control" **type**="number"/>

<**span** **asp-validation-for**="Price" class="text-danger"></**span**>

</div>

</div>

<div class="form-group">

<**label** **asp-for**="Description"></**label**>

<**textarea** **asp-for**="Description" class="form-control"></**textarea**>

</div>

<div class="form-row">

@\* <div class="form-group col-md-4"> \*@

@\* <label asp-for="Quantity"></label> \*@

@\* <input asp-for="Quantity" class="form-control" /> \*@

@\* <span asp-validation-for="Quantity" class="text-danger"></span> \*@

@\* </div> \*@

<div class="form-group col-md-4">

<**label** **asp-for**="EAN"></**label**>

<**input** **asp-for**="EAN" class="form-control" />

<**span** **asp-validation-for**="EAN" class="text-danger"></**span**>

</div>

<div class="form-group col-md-4">

<**label** **asp-for**="SKU"></**label**>

<**input** **asp-for**="SKU" class="form-control" />

<**span** **asp-validation-for**="SKU" class="text-danger"></**span**>

</div>

</div>

<div class="form-group">

<**label** **asp-for**="SupplierId">Dostawca</**label**>

<**select** **asp-for**="SupplierId"

**asp-items**="ViewBag.SupplierList"

class="form-control">

<option value="">— (brak) —</option>

</select>

</div>

<div class="form-group">

<label asp-for="CategoryId">Kategoria</label>

<select asp-for="CategoryId"

asp-items="ViewBag.CategoryList"

class="form-control">

<option value="">— wybierz —</option>

</select>

<span asp-validation-for="CategoryId" class="text-danger"></span>

</div>

@\*

Dwa przyciski:

- jeden o nazwie "ShowFeatures" (przeładowuje widok cech),

- drugi o nazwie "SaveInstrument" (zapisuje do bazy).

\*@

<div class="mb-2 mt-4">

<button type="submit"

name="action"

value="ShowFeatures"

class="btn btn-secondary">

Pokaż cechy

</button>

<button type="submit"

name="action"

value="SaveInstrument"

class="btn btn-primary">

Zapisz instrument

</button>

</div>

</div>

<div class="col-md-6">

@\* Jeżeli w ViewBag jest lista FeatureDefinitions, wyświetlamy tabelę cech \*@

@if (ViewBag.FeatureDefinitions is List<FeatureDefinition> feats && feats.Any())

{

<h4 class="mt-4">Cechy szczegółowe</h4>

<table class="table table-sm">

<thead>

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < feats.Count; i++)

{

<tr>

<td>

@feats[i].Name

@\* ukryte pole z FeatureDefinitionId \*@

<input type="hidden"

name="InstrumentFeatures[@i].FeatureDefinitionId"

value="@feats[i].FeatureDefinitionId" />

</td>

<td>

<input type="text"

name="InstrumentFeatures[@i].Value"

class="form-control" />

</td>

</tr>

}

</tbody>

</table>

}

</div>

</div>

</**form**>

<div>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

### Delete

@model Music\_Store\_Warehouse\_App.Models.Instrument

@{

ViewData["Title"] = "Delete";

}

<h1>Usuwanie</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Instrument</h4>

<hr />

<div class="container">

<div class="row">

<div class="col-md-6">

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Price)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Price)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Description)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Description)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.EAN)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.EAN)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.SKU)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.SKU)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Supplier)

</dt>

<dd class="col-sm-10">

@(Model.Supplier?.Name ?? "Brak przypisanego dostawcy") <!-- Zabezpieczenie przed supplier = null-->

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Category)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Category.Name)

</dd>

</dl>

</div>

<!-- Prawa kolumna: tabela z cechami instrumentu -->

<div class="col-md-6">

<div class="card mb-4">

<div class="card-header bg-light">

<strong>Cechy szczegółowe</strong>

</div>

<div class="card-body p-0">

@if (Model.InstrumentFeatures != null && Model.InstrumentFeatures.Any())

{

<table class="table table-striped mb-0">

<thead class="thead-light">

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@foreach (var feat in Model.InstrumentFeatures)

{

<tr>

<td>

@feat.FeatureDefinition?.Name

</td>

<td>

@feat.Value

</td>

</tr>

}

</tbody>

</table>

}

else

{

<div class="p-3">

<em>Brak przypisanych cech dla tego instrumentu.</em>

</div>

}

</div>

</div>

</div>

</div>

</div>

<**form** **asp-action**="Delete">

<**input** **type**="hidden" **asp-for**="InstrumentId" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</**form**>

</div>

### Details – szczegóły instrumentu

@model Music\_Store\_Warehouse\_App.Models.Instrument

@{

ViewData["Title"] = "Szczegóły";

}

<h1>Szczegóły</h1>

<h4>Instrument</h4>

<hr />

<div class="container">

<div class="row">

<div class="col-md-6">

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Price)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Price)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Description)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Description)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.EAN)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.EAN)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.SKU)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.SKU)

</dd>

@\* <dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Inventory.Quantity)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Inventory.Quantity)

</dd> \*@

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Supplier)

</dt>

<dd class="col-sm-10">

@if (Model.Supplier != null)

{

<!-- Przekierowanie do szczegółów dostawcy za pomocą route-id-->

<**a** **asp-controller**="Suppliers" **asp-action**="Details" **asp-route-id**="@Model.Supplier.SupplierId">

@Model.Supplier.Name

</**a**>

}

else

{

<text>Brak przypisanego dostawcy</text>

}

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Category)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Category.Name)

</dd>

</dl>

</div>

<!-- Prawa kolumna: tabela z cechami instrumentu -->

<div class="col-md-6">

<div class="card mb-4">

<div class="card-header bg-light">

<strong>Cechy szczegółowe</strong>

</div>

<div class="card-body p-0">

@if (Model.InstrumentFeatures != null && Model.InstrumentFeatures.Any())

{

<table class="table table-striped mb-0">

<thead class="thead-light">

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@foreach (var feat in Model.InstrumentFeatures)

{

<tr>

<td>

@feat.FeatureDefinition?.Name

</td>

<td>

@feat.Value

</td>

</tr>

}

</tbody>

</table>

}

else

{

<div class="p-3">

<em>Brak przypisanych cech dla tego instrumentu.</em>

</div>

}

</div>

</div>

</div>

</div>

</div>

<div>

@\* Tymczasowo wyłączone po stronie użytkownika \*@

@\* <a asp-action="Edit" asp-route-id="@Model?.InstrumentId" class="btn btn-sm btn-primary">Edit</a> | \*@

<**a** **asp-controller**="Documents" **asp-action**="Create" **asp-route-supplierId**="Model.SupplierId" **asp-route-selectedInstrumentIds**="Model.InstrumentId" class="btn btn-sm btn-outline-success">Przyjmij do magazynu</**a**>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

### Edit – edycja instrumentu

@model Music\_Store\_Warehouse\_App.Models.Instrument

@{

ViewData["Title"] = "Edycja";

}

<h1>Edycja</h1>

<h4>Instrument</h4>

<hr />

<**form** **asp-action**="Edit">

<div class="row">

<div class="col-md-4">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<**input** **type**="hidden" **asp-for**="InstrumentId" />

<div class="form-group">

<**label** **asp-for**="Name" class="control-label"></**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Price" class="control-label"></**label**>

<**input** **asp-for**="Price" class="form-control" **type**="number" />

<**span** **asp-validation-for**="Price" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Description" class="control-label"></**label**>

<**input** **asp-for**="Description" class="form-control" />

<**span** **asp-validation-for**="Description" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="EAN" class="control-label"></**label**>

<**input** **asp-for**="EAN" class="form-control" />

<**span** **asp-validation-for**="EAN" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="SKU" class="control-label"></**label**>

<**input** **asp-for**="SKU" class="form-control" />

<**span** **asp-validation-for**="SKU" class="text-danger"></**span**>

</div>

@\* <div class="form-group">

<label asp-for="Inventory.Quantity" class="control-label"></label>

<input asp-for="Inventory.Quantity" class="form-control" />

<span asp-validation-for="Inventory.Quantity" class="text-danger"></span> \*@

</div>

<div class="form-group">

<**label** **asp-for**="SupplierId">Dostawca</**label**>

<**select** **asp-for**="SupplierId"

**asp-items**="ViewBag.SupplierList"

class="form-control">

<option value="">— (brak) —</option>

</select>

</div>

<div class="form-group">

<label asp-for="CategoryId">Kategoria</label>

<select asp-for="CategoryId"

asp-items="ViewBag.CategoryList"

class="form-control">

<option value="">— wybierz —</option>

</select>

<span asp-validation-for="CategoryId" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary mt-3 mb-2" />

</div>

</div>

<div class="col-md-2">

<!-- rozdzielacz -->

</div>

<div class="col-md-6">

<!-- Tabela z cechami instrumentu -->

@if (Model.InstrumentFeatures != null && Model.InstrumentFeatures.Any())

{

var featuresList = Model.InstrumentFeatures.ToList();

<h4>Przypisane cechy</h4>

<table class="table table-bordered">

<thead>

<tr>

<th>Nazwa cechy</th>

<th>Wartość</th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < featuresList.Count; i++)

{

// Pojedyncza cecha z listy

var ifeat = featuresList[i];

<tr>

<td>

@ifeat.FeatureDefinition.Name

<!-- Ukryte pola, żeby ASP.NET wiedział, który InstrumentFeature edytujemy -->

<input type="hidden"

name="InstrumentFeatures[@i].InstrumentFeatureId"

value="@ifeat.InstrumentFeatureId" />

<input type="hidden"

name="InstrumentFeatures[@i].FeatureDefinitionId"

value="@ifeat.FeatureDefinitionId" />

</td>

<td>

<!-- Ręcznie składamy pole z name="InstrumentFeatures[i].Value" -->

<input type="text"

name="InstrumentFeatures[@i].Value"

value="@ifeat.Value"

class="form-control" />

</td>

</tr>

}

</tbody>

</table>

}

else

{

<p>Ten instrument nie ma jeszcze przypisanych cech.</p>

}

</div>

</div>

</**form**>

<div>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

### Index – widok wszystkich instrumentów w sieci dystrybucyjnej

@using Music\_Store\_Warehouse\_App.Data

@model PaginatedList<Music\_Store\_Warehouse\_App.Models.Instrument>

@{

ViewData["Title"] = "Instrumenty";

}

<h1>Instrumenty dostępne u dystrybutorów</h1>

<p>

<p>(Dodaj instrumenty do sieci)</p>

<**a** **asp-action**="Create" class="btn btn-sm btn-success">Dodaj</**a**>

</p>

<**form** **asp-action**="Index" method="get" class="form-inline mb-3">

<p class="mb-0">

<!-- Pole wyszukiwania -->

<label class="mr-2">

Szukana nazwa:

<input type="text"

name="searchString"

value="@(ViewData["CurrentFilter"] ?? "")"

class="form-control form-control-sm ml-1"

style="width: 200px;" />

</label>

<!-- Dropdown kategorii -->

<label class="ml-3 mr-2">

kategoria:

<select name="categoryId"

id="categoryId"

class="form-control form-control-sm ml-1"

style="width: 180px;">

<**option** **value**="">Wszystkie</**option**>

@foreach (SelectListItem cat in (SelectList)ViewData["CategoryList"])

{

<**option** **value**="@cat.Value" selected="@(cat.Selected ? "selected" : null)">

@cat.Text

</**option**>

}

</select>

</label>

<!-- Dropdown dostawców -->

<label class="ml-3 mr-2">

dostawca:

<select name="supplierId"

id="supplierId"

class="form-control form-control-sm ml-1"

style="width: 180px;">

<**option** **value**="">Wszyscy</**option**>

@foreach (SelectListItem sup in (SelectList)ViewData["SupplierList"])

{

<option value="@sup.Value" selected="@(sup.Selected ? "selected" : null)">

@sup.Text

</option>

}

</select>

</label>

<!-- Przycisk wyszukiwania i link do pełnej listy -->

<button type="submit" class="btn btn-default">Szukaj</button> |

<a asp-action="Index">Wyczyść</a>

</p>

</form>

<table class="table">

<thead>

<tr>

<th>

<a asp-action="Index" asp-route-sortOrder="@ViewData["NameSortParm"]" asp-route-currentFilter="@ViewData["CurrentFilter"]">Name</a>

</th>

<th>

<a asp-action="Index" asp-route-sortOrder="@ViewData["PriceSortParm"]" asp-route-currentFilter="@ViewData["CurrentFilter"]">Price</a>

</th>

<th>

<!-- Dodajemy [0] ponieaż teraz model to PaginatedList która nie ma właściwości Description tak jak wcześniejszy

model instrument.

\*/ -->

@Html.DisplayNameFor(model => model[0].Description)

</th>

<th>

@Html.DisplayNameFor(model => model[0].EAN)

</th>

<th>

@Html.DisplayNameFor(model => model[0].SKU)

</th>

@\* Od wejśca rozszerzonej wersji ilość jest wyświetlana w controlerze magazyny - InstrumentInventoriesControler

<th>

@Html.DisplayNameFor(model => model[0].Inventory.Quantity)

</th> \*@

<th>

@Html.DisplayNameFor(model => model[0].Supplier)

</th>

<th>

@Html.DisplayNameFor(model => model[0].Category)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

@Html.DisplayFor(modelItem => item.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.EAN)

</td>

<td>

@Html.DisplayFor(modelItem => item.SKU)

</td>

@\* <td>

@Html.DisplayFor(modelItem => item.Inventory.Quantity)

</td> \*@

<td>

@Html.DisplayFor(modelItem => item.Supplier.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Category.Name)

</td>

<td style>

@\* Wyłączone po stronie użytkownika, będzie aktywne gdy pojawi się logowanie admina\*@

@\* <a asp-action="Edit" asp-route-id="@item.InstrumentId" class="btn btn-sm btn-primary">Edytuj</a> | \*@

<**a** **asp-action**="Details" **asp-route-id**="@item.InstrumentId" class="btn btn-sm btn-info">Szczegóły</**a**>

<**a** **asp-controller**="Documents" **asp-action**="Create" **asp-route-supplierId**="@item.SupplierId" **asp-route-selectedInstrumentIds**="@item.InstrumentId" class="btn btn-sm btn-outline-success">Przyjmij do magazynu</**a**>

@\* <a asp-action="Delete" asp-route-id="@item.InstrumentId" class="btn btn-sm btn-danger">Usuń</a> \*@

</td>

</tr>

}

</tbody>

</table>

@{

var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";

var nextDisabled = !Model.HasNextPage ? "disabled" : "";

}

<**a** **asp-action**="Index"

**asp-route-sortOrder**="@ViewData["CurrentSort"]"

**asp-route-pageNumber**="@(Model.PageIndex - 1)"

**asp-route-currentFilter**="@ViewData["CurrentFilter"]"

class="btn btn-default @prevDisabled">

Previous

</**a**>

<**a** **asp-action**="Index"

**asp-route-sortOrder**="@ViewData["CurrentSort"]"

**asp-route-pageNumber**="@(Model.PageIndex + 1)"

**asp-route-currentFilter**="@ViewData["CurrentFilter"]"

class="btn btn-default @nextDisabled">

Next

</**a**>

## Suppliers – zarządzanie dostawcami

### Create – tworzenie dostawcy

@model Music\_Store\_Warehouse\_App.Models.Supplier

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>Supplier</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Create" method="post">

<div class="form-group">

<**label** **asp-for**="Name" class="control-label">Nazwa dostawcy</**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Email" class="control-label">Email</**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<fieldset>

<legend>Dane adresowe dostawcy</legend>

<div class="form-group">

<**label** **asp-for**="Address.Street" class="control-label">Ulica</**label**>

<**input** **asp-for**="Address.Street" class="form-control" />

<**span** **asp-validation-for**="Address.Street" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Address.City" class="control-label">Miasto</**label**>

<**input** **asp-for**="Address.City" class="form-control" />

<**span** **asp-validation-for**="Address.City" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Address.PostalCode" class="control-label">Kod pocztowy</**label**>

<**input** **asp-for**="Address.PostalCode" class="form-control" />

<**span** **asp-validation-for**="Address.PostalCode" class="text-danger"></**span**>

</div>

</fieldset>

<button type="submit" class="btn btn-primary mt-3 mb-2">Zapisz dostawcę z adresem</button>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Delete

@model Music\_Store\_Warehouse\_App.Models.Supplier

@{

ViewData["Title"] = "Usuwanie";

}

<h1>Usuwanie</h1>

<h3>Czy na pewno chcesz usunąć instrument ?</h3>

<div>

<h4>Supplier</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Email)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Address.City)

</dd>

</dl>

<**form** **asp-action**="Delete">

<**input** **type**="hidden" **asp-for**="SupplierId" />

<input type="submit" value="Delete" class="btn btn-danger" />

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</**form**>

</div>

### Details – szczegóły dostawcy

@model Music\_Store\_Warehouse\_App.Models.Supplier

@{

ViewData["Title"] = "Szczegóły";

}

<h1>Szczegóły</h1>

<div>

<h4>Dostawca</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Email)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Address.City)

@Html.DisplayFor(model => model.Address.Street)

@Html.DisplayFor(model => model.Address.PostalCode)

</dd>

</dl>

</div>

<div>

<**a** **asp-action**="Edit" **asp-route-id**="@Model?.SupplierId" class="btn btn-sm btn-primary">Edit</**a**> |

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

### Edit

@model Music\_Store\_Warehouse\_App.Models.Supplier

@{

ViewData["Title"] = "Edycja";

}

<h1>Edycja</h1>

<h4>Dostawca</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Edit">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<**input** **type**="hidden" **asp-for**="SupplierId" />

<div class="form-group">

<**label** **asp-for**="Name" class="control-label"></**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Email" class="control-label"></**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<fieldset>

<legend>Dane adresowe dostawcy</legend>

<div class="form-group">

<**label** **asp-for**="Address.Street" class="control-label">Ulica</**label**>

<**input** **asp-for**="Address.Street" class="form-control" />

<**span** **asp-validation-for**="Address.Street" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Address.City" class="control-label">Miasto</**label**>

<**input** **asp-for**="Address.City" class="form-control" />

<**span** **asp-validation-for**="Address.City" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Address.PostalCode" class="control-label">Kod pocztowy</**label**>

<**input** **asp-for**="Address.PostalCode" class="form-control" />

<**span** **asp-validation-for**="Address.PostalCode" class="text-danger"></**span**>

</div>

</fieldset>

<div class="form-group">

<input type="submit" value="Zapisz" class="btn btn-primary mt-3 mb-2" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index" class="btn btn-sm btn-secondary">Powrót</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Index – lista dostawców isntrumentów

@model IEnumerable<Music\_Store\_Warehouse\_App.Models.Supplier>

@{

ViewData["Title"] = "Dostawcy";

}

<h1>Dostawcy</h1>

<p>

<**a** **asp-action**="Create" class="btn btn-sm btn-success">Dodaj</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.Address)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.Address.City)

@Html.DisplayFor(modelItem => item.Address.Street)

</td>

<td>

<**a** **asp-action**="Edit" **asp-route-id**="@item.SupplierId" class="btn btn-sm btn-primary">Edit</**a**> |

<**a** **asp-action**="Details" **asp-route-id**="@item.SupplierId" class="btn btn-sm btn-info">Details</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.SupplierId" class="btn btn-sm btn-danger">Delete</**a**>

</td>

</tr>

}

</tbody>

</table>

## Shared

### \_Layout – ogólny widok dla każdej strony

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - MusicStore\_Warehouse\_App</title>

<**script** **type**="importmap"></**script**>

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<**link** rel="stylesheet" **href**="~/Music\_Store\_Warehouse\_App.styles.css" **asp-append-version**="true" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<**a** class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Strona główna</**a**>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Instruments" **asp-action**="Index">Intrumenty</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Suppliers" **asp-action**="Index">Dostawcy</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="InstrumentInventories" **asp-action**="Index">Magazyn</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Documents" **asp-action**="Index">Dokumenty</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="About">O aplikacji</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Polityka prywatności</**a**>

</li>

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2025 - Music Store Warehouse App - <**a** **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Polityka prywatności</**a**>

</div>

</footer>

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** **src**="~/js/site.js" **asp-append-version**="true"></**script**>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

### error – widok błędu

@model ErrorViewModel

@{

ViewData["Title"] = "Error";

}

<h1 class="text-danger">Error.</h1>

<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)

{

<p>

<strong>Request ID:</strong> <code>@Model.RequestId</code>

</p>

}

<h3>Development Mode</h3>

<p>

Swapping to <strong>Development</strong> environment will display more detailed information about the error that occurred.

</p>

<p>

<strong>The Development environment shouldn't be enabled for deployed applications.</strong>

It can result in displaying sensitive information from exceptions to end users.

For local debugging, enable the <strong>Development</strong> environment by setting the <strong>ASPNETCORE\_ENVIRONMENT</strong> environment variable to <strong>Development</strong>

and restarting the app.

</p>

# 