**Міністерство освіти та науки України**

**Київський національний торговельно-економічний університет Факультет інформаційних технологій**

**Кафедра інженерії програмного забезпечення та кібербезпеки**

**ПРАКТИЧНА РОБОТА  НА ТЕМУ:**

**Інформаційна система MUKILSKO\_INFO**



(назва теми)

Виконав студент факультету

Глижін Владислав Сергійович

Групи 7 курсу 3

Київ 20 22

**Зміст**

[1. Предметна область 3](#_Toc107388587)

[2. Архітектура розроблюваного веб додатку 4](#_Toc107388588)

[2.1. Еталонна модель веб сайту 4](#_Toc107388589)

[2.2. Блок схема веб сайту 5](#_Toc107388590)

[2.3. Use case діаграма 6](#_Toc107388591)

[2.4. Діаграма MVC архітектури 7](#_Toc107388592)

[3. Опис бази даних 8](#_Toc107388593)

[3.1. Логічна модель бази даних 8](#_Toc107388594)

[3.2. Фізична модель бази даних 8](#_Toc107388595)

[4. Нормалізація бази даних 9](#_Toc107388596)

[5. Дизайн проекту у Figma 10](#_Toc107388597)

[6. Додатки 12](#_Toc107388598)

[Додаток А 12](#_Toc107388599)

[Додаток Б 13](#_Toc107388600)

[Додаток В 17](#_Toc107388601)

# **Предметна область**

В Україні починаючи з 2015 року стрімко стали набувати популярності вживані, злегка пошкоджені, однак досить вигідні до придбання, порівнюючи з ринком України, автомобілі з онлайн аукціонів Сполучених Штатів Америки (США) таких як Copart[1], Manheim[2], та IAA-Insurance[3].

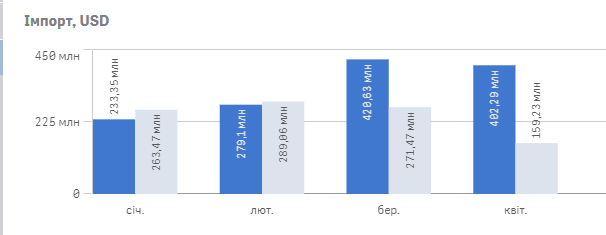
Як важливий факт, слід відзначити тенденцію зниження цін на вживані автомобілі. Очевидно, що це є наслідком збільшення кількості автомобілей на ринку. Адже лише за березень 2021р., в Україну було ввезено транспортних засобів на 420 млн. доларів. Що є більшою половиною від річної суми за цілий минулий рік рис. 1.2.

Рисунок 1 – Імпорт авто в Україну

Всі ці фактори впливають на розповсюдження на перший погляд ексклюзивних авто.

# **Архітектура розроблюваного веб додатку**

## **Еталонна модель веб сайту**

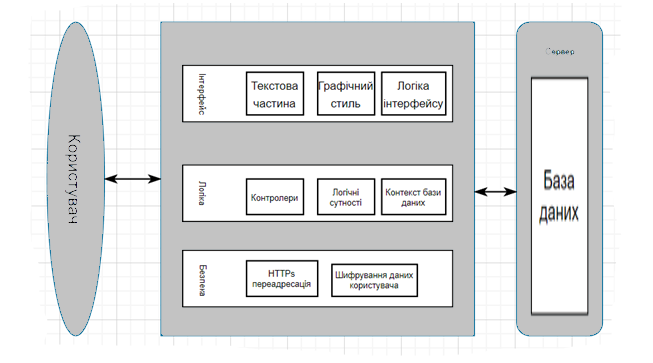


Рисунок 2. Еталонна модель веб сайту

*Джерело: побудовано автором*

## Блок схема веб сайту

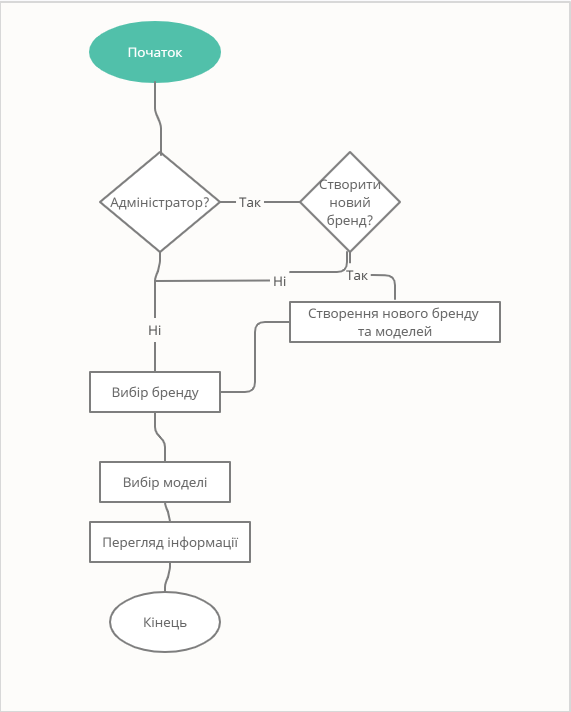


Рисунок 3. Блок схема веб сайту

*Джерело: побудовано автором*

## Use case діаграма

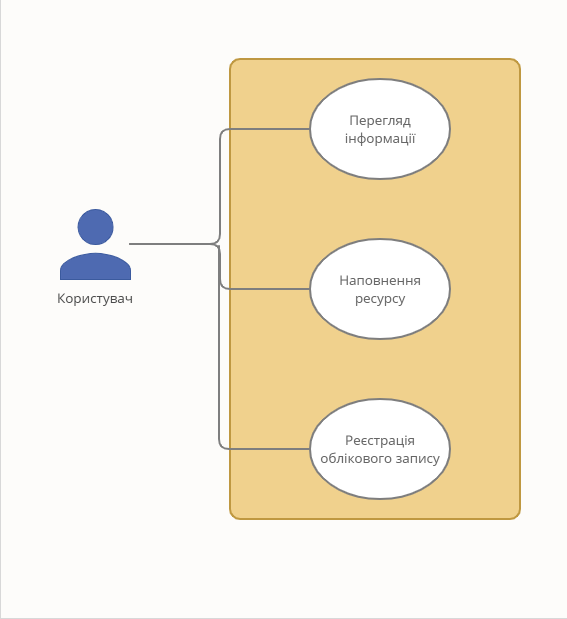


Рисунок 4. Use case діаграма

*Джерело: побудовано автором*

## Діаграма MVC архітектури

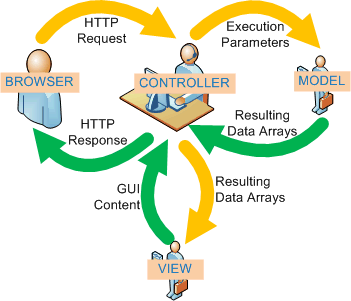


Рисунок 5. Діаграма послідовності

*Джерело: https://www.slideshare.net/surbhiiiii/mvc-architecture-17908772*

Для побудови програми найважливіша частина – це вибір архітектурного паттерну. Для ASP Core найпопулярнішим архітектурним паттерном є MVC(Model-View-Controller). Суть MVC – це розділення даних програми та керуючої логіки на три окремі компоненти: модель, уявлення та контролер — таким чином, що модифікація кожного компонента може здійснюватися незалежно.

# **Опис бази даних**

## Логічна модель бази даних

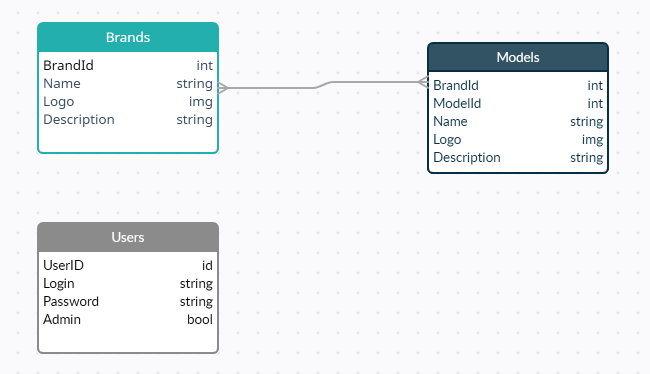


Рисунок 6. Логічна модель бази даних

*Джерело: побудовано автором*

## Фізична модель бази даних

1. Таблиця Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва поля | Ключ | Індексоване поле | Тип даних | Null-значення |
| UserId | Primary key | Так | int | Не допустимі |
| UserName |  |  | nvarchar(256) | Не допустимі |
| Email |  |  | nvarchar(256) | Не допустимі |
| PasswordHash |  |  | nvarchar(MAX) | Не допустимі |

1. Таблиця Brands

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва поля | Ключ | Індексоване поле | Тип даних | Null-значення |
| BrandId | Primary Key |  | int | Не допустимі |
| BrandName |  |  | nvarchar(256) | Не допустимі |
| Logo |  |  | image | Не допустимі |
| Description |  |  | nvarchar(256) | Не допустимі |

1. Таблиця Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва поля | Ключ | Індексоване поле | Тип даних | Null-значення |
| BrandId | Foreign Key |  | int | Не допустимі |
| ModelId | Primary Key |  | int | Не допустимі |
| ModelName |  |  | nvarchar(256) | Не допустимі |
| Logo |  |  | image | Не допустимі |
| Description |  |  | nvarchar(256) | Не допустимі |

## **Нормалізація бази даних**

Нормалізація - це процес організації даних у базі даних, що включає створення таблиць і встановлення відносин між ними відповідно до правил, які забезпечують захист даних і роблять базу даних більш гнучкою, усуваючи надмірність та неузгоджені залежності.

Надмірність даних призводить до непродуктивного витрачання вільного місця на диску та ускладнює обслуговування баз даних. Наприклад, якщо дані, які зберігаються в кількох місцях, потрібно змінити, в них доведеться внести ті самі зміни у всіх цих місцях.

*Приведення бази даних до першої нормальної форми*

Відношення буде зведено до **першої нормальної форми** (**1НФ**) тоді й тільки тоді, коли всі його атрибути містять тільки неподільні (атомарні) значення й у ньому відсутні групи атрибутів з однаковими за змістом значеннями, які повторюються у межах одного кортежу

*Приведення бази даних до другої нормальної форми*

Відношення буде зведено до **другої нормальної форми** (**2НФ**) тоді й тільки тоді, коли воно є в першій нормальній формі, і кожний неключовий атрибут повністю визначається первинним ключем, тобто щоб первинний ключ однозначно визначав кортеж і не був надлишковим (збігався із суперключем). Ті атрибути, які залежать тільки від частини суперключа, мають бути виділені в окремі таблиці.

# **Дизайн проекту у Figma**

Для побудови графічної частини програми, потрібно розробити дизайн, як найкраще для створення прототипу сайту підходить додаток Figma. Figma – це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері. І це лише одне з важливих переваг платформи.



Рисунок 7. Дизайн основної сторінки

*Джерело: побудовано автором (знімок екрану)*

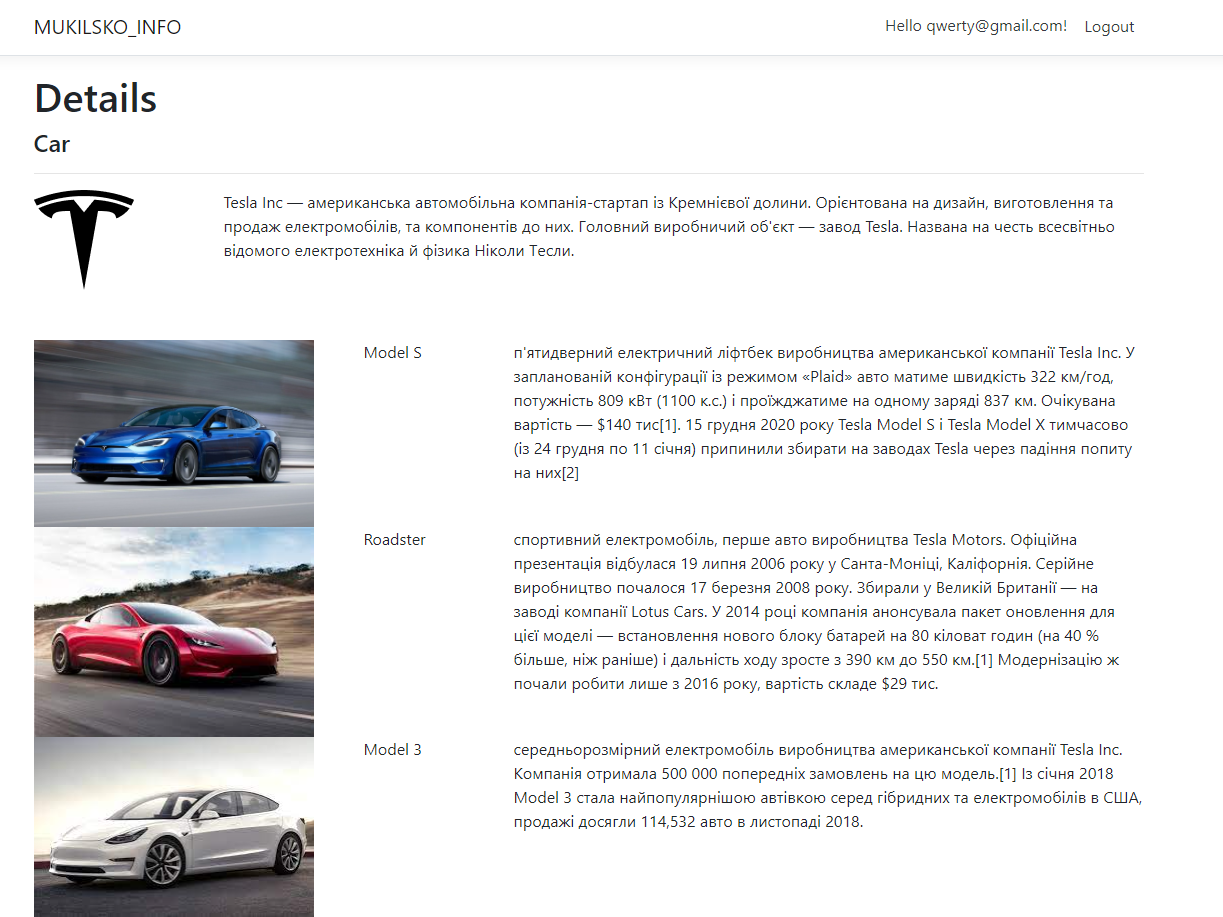


Рисунок 8. Модельний ряд

*Джерело: побудовано автором (знімок екрану)*

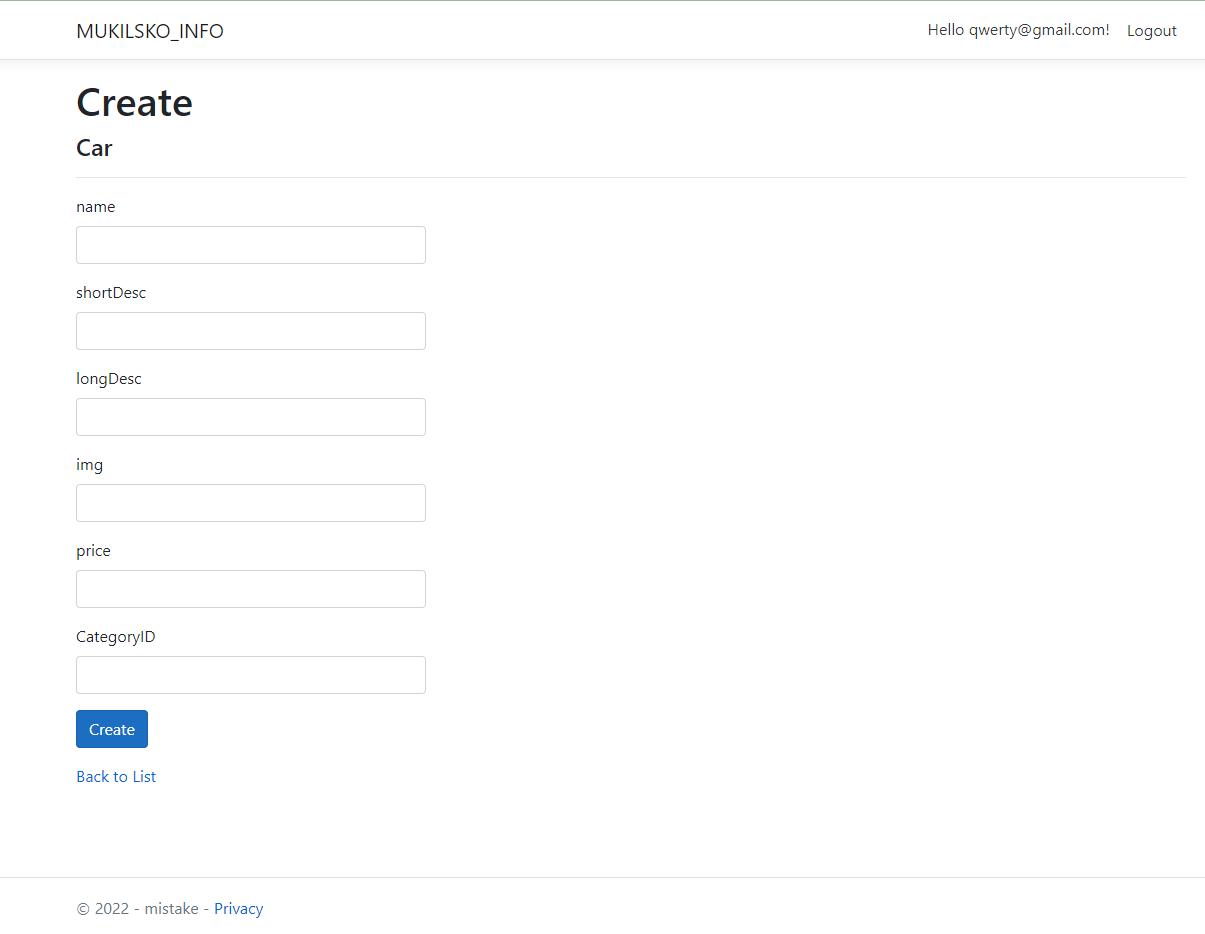


Рисунок 9. Створення Бренду

*Джерело: побудовано автором (знімок екрану)*

# **Додатки**

## Додаток А

Програмний код HomeController.cs

using System;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using mistake.Models;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

namespace mistake.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();

}

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

public IActionResult Error()

{

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

}

}

}

## Додаток Б

Програмний код CarsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using mistake.Data;

using mistake.Models;

namespace mistake

{

[Authorize]

public class CarsController : Controller

{

private readonly ApplicationDbContext \_context;

public CarsController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Cars

public async Task<IActionResult> Index()

{

return View(await \_context.Cars.ToListAsync());

}

// GET: Cars/Details/5

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

{

if (id == null)

{

return NotFound();

}

var car = await \_context.Cars

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

if (car == null)

{

return NotFound();

}

var models = await \_context.Models.Where(x => x.CarId == car.id).ToListAsync();

ViewBag.Models = models;

return View(car);

}

// GET: Cars/Create

public IActionResult Create()

{

return View();

}

// POST: Cars/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("id,name,shortDesc,longDesc,img,price,CategoryID")] Car car)

{

if (ModelState.IsValid)

{

\_context.Add(car);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(car);

}

// GET: Cars/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var car = await \_context.Cars.FindAsync(id);

if (car == null)

{

return NotFound();

}

return View(car);

}

// POST: Cars/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("id,name,shortDesc,longDesc,img,price,CategoryID")] Car car)

{

if (id != car.id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(car);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CarExists(car.id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(car);

}

// GET: Cars/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var car = await \_context.Cars

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

if (car == null)

{

return NotFound();

}

return View(car);

}

// POST: Cars/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var car = await \_context.Cars.FindAsync(id);

\_context.Cars.Remove(car);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CarExists(int id)

{

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

}

}

}

## Додаток В

Програмний код ModelsController.cs

using System.Diagnostics;

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 mistake.Data;

using mistake.Models;

namespace mistake

{

public class ModelsController : Controller

{

private readonly ApplicationDbContext \_context;

public ModelsController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Models

public async Task<IActionResult> Index()

{

return View(await \_context.Models.ToListAsync());

}

// GET: Models/Details/5

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

{

if (id == null)

{

return NotFound();

}

var model = await \_context.Models

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

if (model == null)

{

return NotFound();

}

return View(model);

}

// GET: Models/Create

public IActionResult Create()

{

return View();

}

// POST: Models/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("Id,CarId,Name,Img,Details")] Model model)

{

if (ModelState.IsValid)

{

\_context.Add(model);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Create));

}

return View(model);

}

// GET: Models/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var model = await \_context.Models.FindAsync(id);

if (model == null)

{

return NotFound();

}

return View(model);

}

// POST: Models/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("Id,CarId,Name,Img,Details")] Model model)

{

if (id != model.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(model);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ModelExists(model.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(model);

}

// GET: Models/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var model = await \_context.Models

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

if (model == null)

{

return NotFound();

}

return View(model);

}

// POST: Models/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var model = await \_context.Models.FindAsync(id);

\_context.Models.Remove(model);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool ModelExists(int id)

{

return \_context.Models.Any(e => e.Id == id);

}

}

}