|  |  |  |  |
| --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | |
| ФЕДЕРАЛЬНОЕ государственное АВТОНОМНОЕ образовательное учреждение  высшего образования  «КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» | | | |
| НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ) | | | |
| КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ (ИС) | | | |
|  | | | |
| Утверждаю | | | |
| Заведующий кафедрой ИС | | | |
|  | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.А. Валиев | | | |
|  | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г. | | | |
|  | | | |
|  | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | |
|  | | | |
| по дисциплине: | | | |
| **«Базы данных»** | | | |
| на тему: | | | |
| **«** **Разработка информационной системы**  **на основе MS SQL Server »** | | | |
|  | | | |
| (Вариант: АИС для магазина автозапчастей) | | | |
|  | | | |
|  | | | |
| Автор: | | Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| студент группы 2181121 | |  | |
|  | | Руководитель: к.т.н. | |
| \_\_\_\_\_\_\_\_\_\_\_\_ | Багаутдинов Б.И. | доцент кафедры ИС | |
|  |  |  |  |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_ | Ш.Ш. Хузятов |
|  |  |  | |
|  |  | Дата защиты:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г. | |
|  |  |  |  |
|  | | | |
|  | | | |
|  | | | |
| Набережные Челны | | | |
| 2021 | | | |

|  |
| --- |
| КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ |
| НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ) |
| КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ (ИС) |
|  |
|  |
| Утверждаю |
| Заведующий кафедрой ИС |
|  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.А. Валиев |
|  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г. |
|  |
| **ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ** |
|  |
| Студент: Багаутдинов Б.И. |
|  |
| 1 Тема |
| «Разработка информационной системы на основе MS SQL Server» |
| 2 Срок представления к защите |
|  |
| 3 Исходные данные |
| Анализ предметной области. |
| 4 Перечень подлежащих разработке вопросов  Анализ технологий создания веб-проектов, проектирование автоматизированной информационной системы, реализация автоматизированной информационной системы. |

|  |  |  |  |
| --- | --- | --- | --- |
| Задание выдано | \_\_\_\_\_\_\_\_\_\_\_\_г. | \_\_\_\_\_\_\_\_\_\_ | Ш.Ш. Хузятов |
|  |  |  |  |
| Задание принято | \_\_\_\_\_\_\_\_\_\_\_\_г. | \_\_\_\_\_\_\_\_\_\_ | Б.И. Багаутдинов |

Содержание

**Элементы оглавления не найдены.**

# 1 Разработка пользовательского интерфейса

Интерфейс одна из важнейших концепция приложения при его разработке, соответственно ему должно быть уделено существенное внимание. При его разработке важно учитывать ряд параметров, которым он обязательно должен соответствовать, а именно:

* элементы управления системой;
* навигация между блоками системы;
* средства отображения информации;
* устройство и технологию ввода данных;
* обратная связь с пользователем;
* набор задач пользователя, решаемых при помощи данной системы.

Это основные общие требования предъявляемые к любым системам, остальные требования формируются исходя из предметной области.

Интерфейс главной страницы выглядит следующим образом.

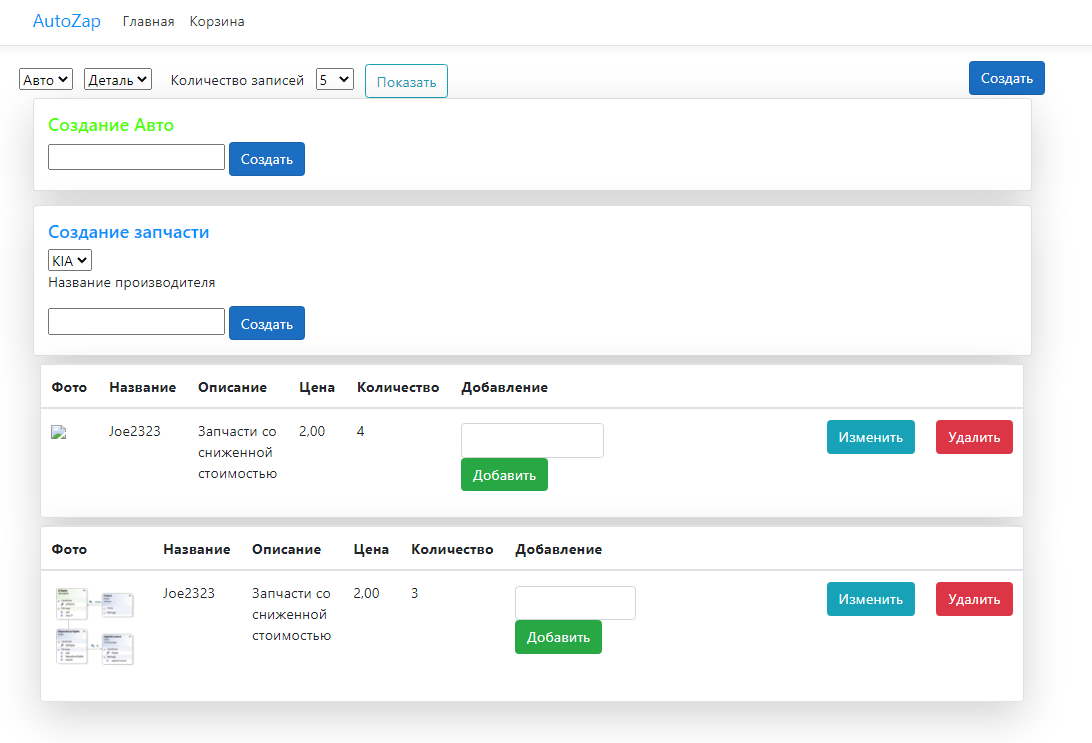


Рисунок 1.1 – Интерфейс главной страницы

На нем реализовано несколько концептуальных блоков, отвечающих за навигацию, фильтрацию, создание и вывод товаров.

Блок навигации предельно прост и реализован в шапке веб-сайта. При клике по соответствующей надписи пользователь перейдет на нужную ему страницу.

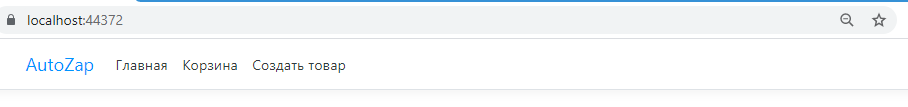


Рисунок 1.2 – Блок навигации

Следующий блок фильтрации товаров. Он предоставляет возможность фильтрации товаров по марке автомобиля и по объединённой группе запчастей. Так же реализована функции вывода определенного количества записей.

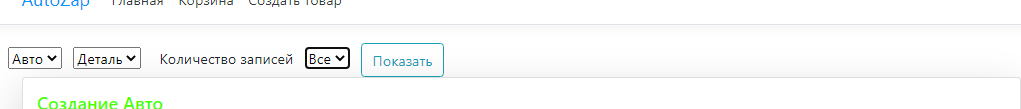


Рисунок 1.3 – Блок фильтрации

Блок создания марок автомобиля и групп запчастей выглядит следующим образом.

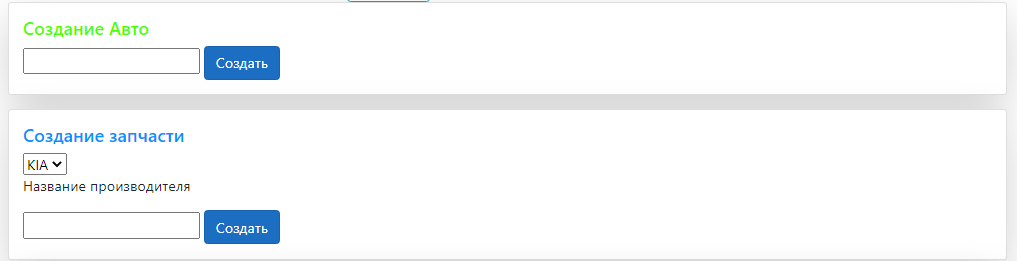


Рисунок 1.4 – Блок создания

Далее следует таблица с товарами, куда выводятся товары по заданным параметрам фильтрации.

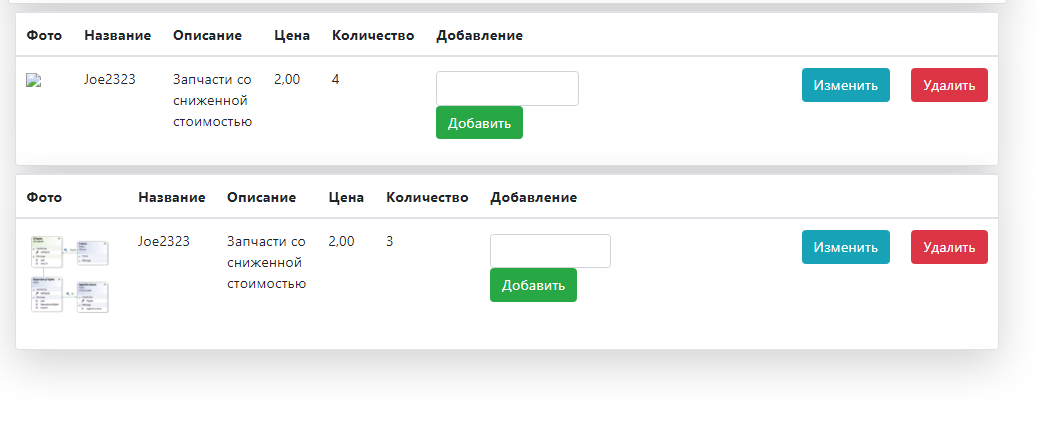


Рисунок 1.5 – Таблица товаров

Так же в ней возможно, перейти к редактированию конкретного товара, добавлению его в корзину или удалить его.

Интерфейс корзины сформирован на основе главной страницы, с редактированием некоторых блоков и добавления новый. Добавлена информация о сумме заказа и кнопка для оформления заказа.

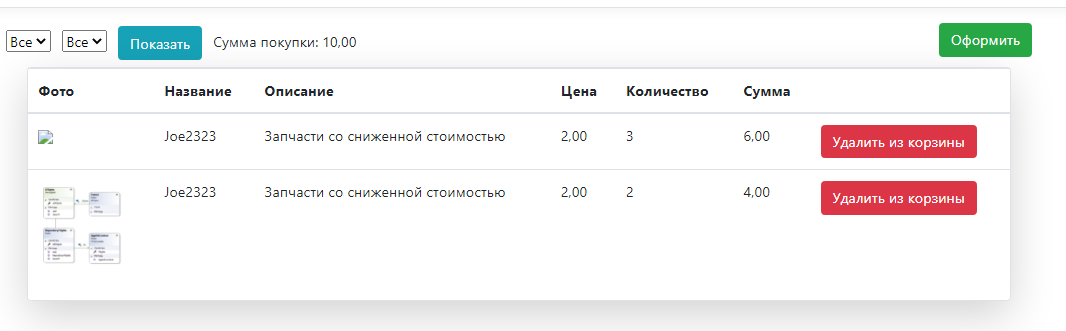


Рисунок 1.6 - Корзина

Через панель навигации можно перейти к созданию нового товара. Интерфейс создания товара выглядит следующим образом. Поля для заполнения, выпадающие списки для выбора марки и группы, а так же загрузка фото.

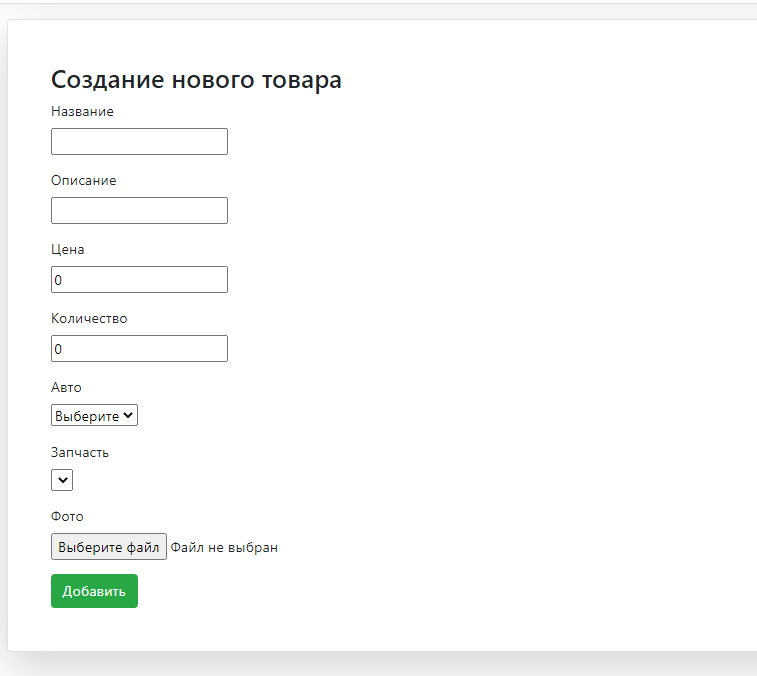


Рисунок 1.7 – Создание товара

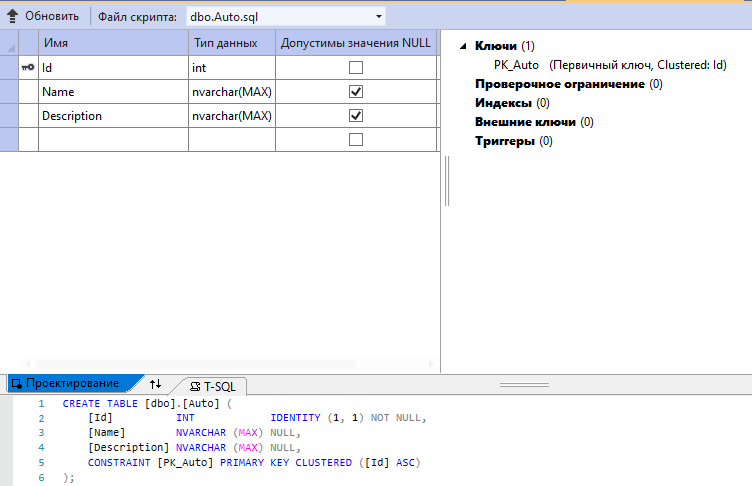
# 2 Проектирование базы данных

Базы данных используются для хранения информации, которая требуется для работы информационной системы. Все данные о товарах, категориях и тому подобных хранятся в ней. Поэтому база данных является одним из важнейших элементов в работе автоматизированной информационной системы.

Проектирование подразумевает под собой определение таблиц, по определенному признаку, полей в этих таблицах и выстраивание связей между таблицами.

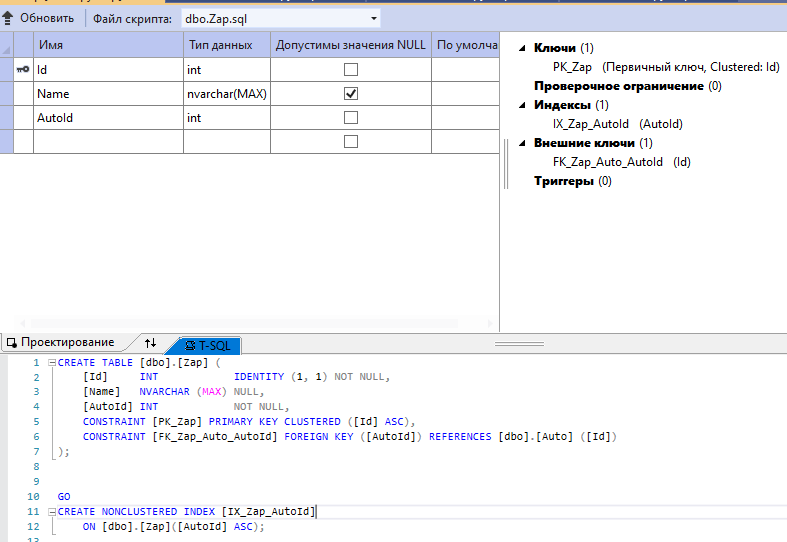
Для этого был произведен анализ предметной области. Исходя из этого и была спроектирована следующая база данных.

Таблица 1. Марки Авто



В данной таблице содержится информация о марках авто, которые есть в информационной системе.

Таблица 2. Группы запчастей



Данная таблица связана внешним ключом с таблицей марок авто, так как для каждой марки авто создается свой категория запчастей.

Таблица 3. Товары

Таблица товаров связана с двумя таблицами – марок авто и групп запчастей. Тип отношения один ко многим.

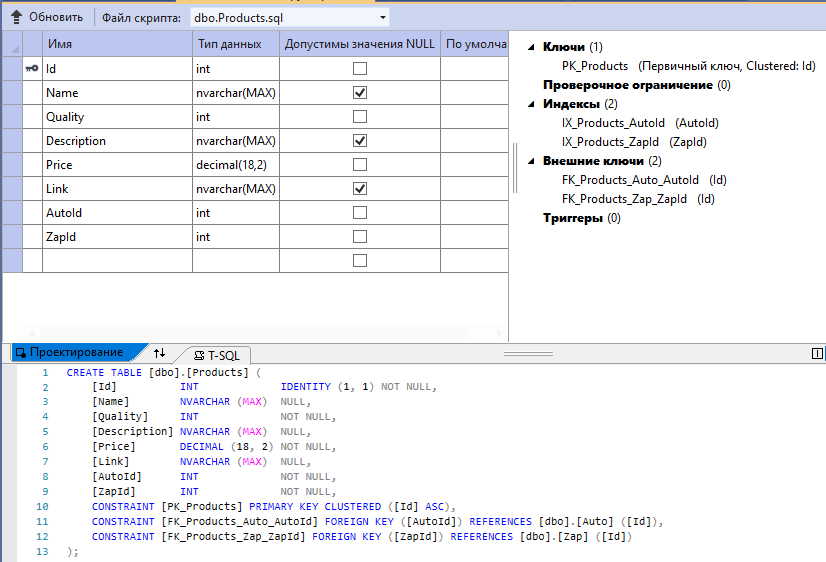
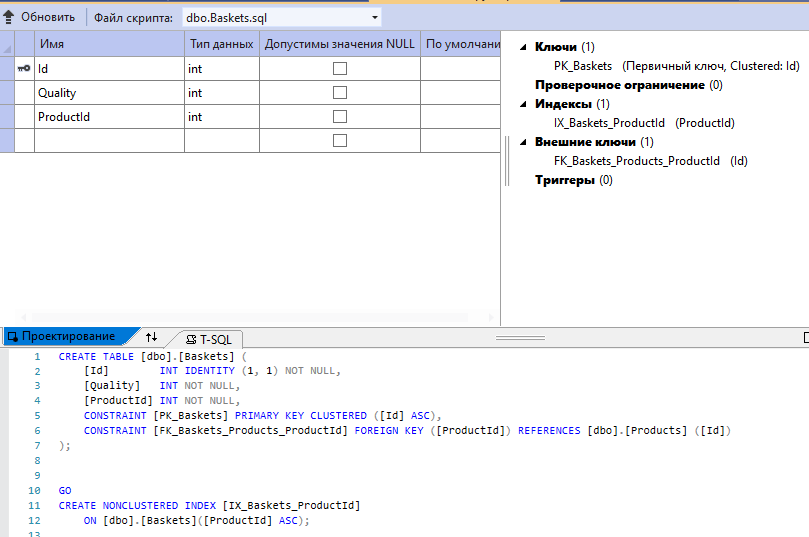


Таблица 4. Корзина



Данная таблица связана внешним ключом с таблицей товаров. Тип отношения один к одному.

# 3 Описание кода программы

Код контроллера отвечающего за формирование данных на главной странице. При чем для фильтрации не используется отдельный метод, весь функционал реализован в одном.

public IActionResult Index(int? catId, int? manId,IndexViewModel model1)

{

var product = db.Products.AsQueryable();

if (catId != null)

{

product = product.Where(x => x.AutoId == catId);

}

if (manId != null)

{

product = product.Where(x => x.ZapId == manId);

}

if(model1.Count == 0)

{

IndexViewModel model2 = new IndexViewModel

{

Products = product.ToList(),

Category = db.Auto.ToList(),

};

return View(model2);

}

IndexViewModel model = new IndexViewModel

{

Products = product.Take(model1.Count).ToList(),

Category = db.Auto.ToList(),

};

return View(model);

}

Создание марки авто и групп запчастей реализовано следующим образом.

[HttpPost]

public IActionResult CreateCat(IndexViewModel model)

{

var cat = new Auto

{

Name = model.CreateViewModel.NewCat

};

db.Auto.Add(cat);

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpPost]

public IActionResult CreateMan(IndexViewModel model)

{

var man = new Zap

{

Name = model.CreateViewModel.NewMan,

AutoId = model.CreateViewModel.CatId

};

db.Zap.Add(man);

db.SaveChanges();

return RedirectToAction("Index");

}

За создание товара отвечает HomeCotroller.cs метод создания имеет валидацию данных при их получении, для того, чтобы записать в базу ошибочные данные и не сломать работу всей системы.

[HttpPost]

public async Task<IActionResult> Create(CreateViewModel model)

{

string path = " ";

if (model.UploadedFile != null)

{

// путь к папке Files

path = "/Files/" + model.UploadedFile.FileName;

// сохраняем файл в папку Files в каталоге wwwroot

using (var fileStream = new FileStream(\_appEnvironment.WebRootPath + path, FileMode.Create))

{

await model.UploadedFile.CopyToAsync(fileStream);

}

}

Products product = new Products

{

Name = model.Name,

Description = model.Description,

AutoId = model.CatId,

ZapId = model.ManId,

Price = model.Price,

Quality = model.Quality,

Link = path

};

db.Products.Add(product);

db.SaveChanges();

return RedirectToAction("Index");

}

Следующее одно из часто выполняющихся действий в информационной системе это добавление товара в корзине, код обработчика приведен ниже.

public IActionResult Add(int id, int quantity)

{

var product = db.Products.FirstOrDefault(x => x.Id == id);

if(product == null)

{

return RedirectToAction("Index", "Home");

}

var basket = new Basket();

basket.ProductId = id;

if(quantity > product.Quality)

{

basket.Quality = product.Quality;

}

else

{

basket.Quality = quantity;

}

product.Quality = product.Quality - basket.Quality;

db.Products.Update(product);

db.Baskets.Add(basket);

db.SaveChanges();

return RedirectToAction("Index", "Home");

}

При удалении товара, так же нужна валидация, чтобы в ответ на запрос к базе данных не получить ошибку.

public IActionResult Delete(int id)

{

var basket = db.Baskets.FirstOrDefault(x => x.ProductId == id);

if(basket == null)

{

return RedirectToAction("Index");

}

var product = db.Products.FirstOrDefault(x => x.Id == id);

product.Quality += basket.Quality;

db.Baskets.Remove(basket);

db.Products.Update(product);

db.SaveChanges();

return RedirectToAction("Index");

}

# 4 Список использованной литературы

1. Bright P. .NET Core 1.0 released, now officially supported by Red Hat: And Samsung is joining the .NET Foundation. — 2016.
2. core/LICENSE.TXT at master · dotnet/core
3. Руководство по .NET Core
4. A Brief History of .NET Core
5. Announcing .NET Core 2.0. .NET Blog (14 августа 2017).
6. Lander, Rich .NET Core 3 and Support for Windows Desktop Applications. MSDN. Microsoft (7 мая 2018).
7. Announcing .NET Core 3.1. .NET Blog (3 декабря 2019).
8. Introducing .NET 5 (англ.). .NET Blog (6 May 2019). Дата обращения 9 февраля 2020.
9. Introduction to .NET Core

# 5 Приложение

Контроллеры

ApiController.cs

[Route("api/[controller]")]

[ApiController]

public class ApiController:ControllerBase

{

ApplicationDbContext db;

public ApiController(ApplicationDbContext db)

{

this.db = db;

}

[Route("GetMan")]

[HttpGet]

public async Task<IActionResult> GetMan(int catId)

{

var result = db.Zap.Where(x => x.AutoId == catId).AsQueryable();

return Ok(result);

}

}

BaskerCotroller.cs

public class BasketController:Controller

{

ApplicationDbContext db;

public BasketController(ApplicationDbContext db)

{

this.db = db;

}

public IActionResult Index(int? catId, int? manId)

{

var ids = db.Baskets.Where(x => x.Quality != 0).Select(x => x.ProductId).ToList();

var product = db.Products.Where(x => ids.Contains(x.Id));

if (catId != null)

{

product = product.Where(x => x.AutoId == catId);

}

if (manId != null)

{

product = product.Where(x => x.ZapId == manId);

}

IndexViewModel model = new IndexViewModel

{

Products = product.ToList(),

Category = db.Auto.ToList(),

};

decimal buySum = 0;

foreach (var t in model.Products)

{

var bas = db.Baskets.FirstOrDefault(x => x.ProductId == t.Id);

t.Quality = bas.Quality;

buySum += t.Quality \* t.Price;

}

ViewBag.Sum = buySum;

return View(model);

}

public IActionResult Add(int id, int quantity)

{

var product = db.Products.FirstOrDefault(x => x.Id == id);

if(product == null)

{

return RedirectToAction("Index", "Home");

}

var basket = new Basket();

basket.ProductId = id;

if(quantity > product.Quality)

{

basket.Quality = product.Quality;

}

else

{

basket.Quality = quantity;

}

product.Quality = product.Quality - basket.Quality;

db.Products.Update(product);

db.Baskets.Add(basket);

db.SaveChanges();

return RedirectToAction("Index", "Home");

}

public IActionResult Delete(int id)

{

var basket = db.Baskets.FirstOrDefault(x => x.ProductId == id);

if(basket == null)

{

return RedirectToAction("Index");

}

var product = db.Products.FirstOrDefault(x => x.Id == id);

product.Quality += basket.Quality;

db.Baskets.Remove(basket);

db.Products.Update(product);

db.SaveChanges();

return RedirectToAction("Index");

}

public IActionResult Buy()

{

var basket = db.Baskets.ToList();

db.Baskets.RemoveRange(basket);

db.SaveChanges();

return RedirectToAction("Index");

}

}

HomeController.cs

public class HomeController : Controller

{

private ApplicationDbContext db;

IWebHostEnvironment \_appEnvironment;

public HomeController(ApplicationDbContext db, IWebHostEnvironment appEnvironment)

{

this.db = db;

\_appEnvironment = appEnvironment;

}

public IActionResult Index(int? catId, int? manId,IndexViewModel model1)

{

var product = db.Products.AsQueryable();

if (catId != null)

{

product = product.Where(x => x.AutoId == catId);

}

if (manId != null)

{

product = product.Where(x => x.ZapId == manId);

}

if(model1.Count == 0)

{

IndexViewModel model2 = new IndexViewModel

{

Products = product.ToList(),

Category = db.Auto.ToList(),

};

return View(model2);

}

IndexViewModel model = new IndexViewModel

{

Products = product.Take(model1.Count).ToList(),

Category = db.Auto.ToList(),

};

return View(model);

}

[HttpGet]

public IActionResult Create()

{

CreateViewModel model = new CreateViewModel

{

Category = db.Auto.ToList()

};

return View(model);

}

[HttpGet]

public IActionResult CreateCat()

{

CreateViewModel model = new CreateViewModel

{

Category = db.Auto.ToList()

};

return View(model);

}

[HttpPost]

public IActionResult CreateCat(IndexViewModel model)

{

var cat = new Auto

{

Name = model.CreateViewModel.NewCat

};

db.Auto.Add(cat);

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpPost]

public IActionResult CreateMan(IndexViewModel model)

{

var man = new Zap

{

Name = model.CreateViewModel.NewMan,

AutoId = model.CreateViewModel.CatId

};

db.Zap.Add(man);

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpPost]

public async Task<IActionResult> Create(CreateViewModel model)

{

string path = " ";

if (model.UploadedFile != null)

{

// путь к папке Files

path = "/Files/" + model.UploadedFile.FileName;

// сохраняем файл в папку Files в каталоге wwwroot

using (var fileStream = new FileStream(\_appEnvironment.WebRootPath + path, FileMode.Create))

{

await model.UploadedFile.CopyToAsync(fileStream);

}

}

Products product = new Products

{

Name = model.Name,

Description = model.Description,

AutoId = model.CatId,

ZapId = model.ManId,

Price = model.Price,

Quality = model.Quality,

Link = path

};

db.Products.Add(product);

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpGet]

public IActionResult Edit(int id)

{

var prod = db.Products.Find(id);

EditViewModel model = new EditViewModel

{

Id = prod.Id,

Name = prod.Name,

Description = prod.Description,

Price = prod.Price,

Quality = prod.Quality

};

return View(model);

}

[HttpPost]

public async Task<IActionResult> Edit(EditViewModel model)

{

var prod = db.Products.Find(model.Id);

prod.Name = model.Name;

prod.Description = model.Description;

prod.Price = model.Price;

prod.Quality = model.Quality;

string path;

if (model.File != null)

{

// путь к папке Files

path = "/Files/" + model.File.FileName;

// сохраняем файл в папку Files в каталоге wwwroot

using (var fileStream = new FileStream(\_appEnvironment.WebRootPath + path, FileMode.Create))

{

await model.File.CopyToAsync(fileStream);

}

prod.Link = path;

}

db.Products.Update(prod);

db.SaveChanges();

return RedirectToAction("Index");

}

}

ApplicationDbContext.cs

public class ApplicationDbContext:DbContext

{

public DbSet<Products> Products { get; set; }

public DbSet<Auto> Auto { get; set; }

public DbSet<Basket> Baskets { get; set; }

public DbSet<Zap> Zap { get; set; }

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

}

CreateViewModel.cs

public class CreateViewModel

{

public List<Auto> Category { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public int CatId { get; set; }

public string NewCat { get; set; }

public int ManId { get; set; }

public string NewMan { get; set; }

public int Price { get; set; }

public int Quality { get; set; }

public IFormFile UploadedFile { get; set; }

}

EditViewModel.cs

public class EditViewModel

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public int Quality { get; set; }

public IFormFile File { get; set; }

}

IndexViewModel.cs

public class IndexViewModel

{

public List<Products> Products { get; set; }

public List<Auto> Category { get; set; }

public CreateViewModel CreateViewModel { get; set; }

public int Count { get; set; }

}

Модели таблиц

Auto.cs

public class Auto

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public virtual ICollection<Products> Products { get; set; }

public virtual ICollection<Zap> Zap { get; set; }

}

Basket.cs

public class Basket

{

public int Id { get; set; }

public int Quality { get; set; }

[ForeignKey("Products")]

public int ProductId { get; set; }

public Products Products { get; set; }

}

Products.cs

public class Products

{

public int Id { get; set; }

public string Name { get; set; }

public int Quality { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public string Link { get; set; }

[ForeignKey("Auto")]

public int AutoId { get; set; }

public virtual Auto Category { get; set; }

[ForeignKey("Zap")]

public int ZapId { get; set; }

public virtual Zap Zap { get; set; }

public ICollection<Basket> Baskets { get; set; }

}

Zap.cs

public class Zap

{

[Key]

public int Id { get; set; }

public string Name { get; set; }

public ICollection<Products> Products { get; set; }

[ForeignKey("Auto")]

public int AutoId { get; set; }

}

Представления

Basket Index.cshtml

@model MobilePhoneBD.Models.ViewMode.IndexViewModel

@{

ViewData["Title"] = "корзина";

}

<div class="text-center row justify-content-between">

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

<select name="catId" onchange="GetMan()" id="cat">

<option **value**="0">Все</option>

@foreach (var cat in Model.Category)

{

<option **value**="@cat.Id">@cat.Name</option>

}

</select>

<select name="manId" id="man" class="m-2">

<option **value**="0">Все</option>

</select>

<button type="submit" class="btn btn-info">Показать</button>

<label class="ml-2">Сумма покупки:</label>

<label id="buySum">@ViewBag.Sum</label>

</form>

@if (Model.Products.Count != 0)

{

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

<input type="submit" value="Оформить" class="btn btn-success" />

</form>

}

</div>

<div>

<div class="card border shadow-lg m-2">

<table class="table">

<thead><tr><th>Фото</th><th>Название</th><th>Описание</th><th>Цена</th><th>Количество</th><th>Сумма</th><th></th></tr></thead>

<tbody>

@foreach (var prod in Model.Products)

{

<tr>

<td><img style="width:100px" src="@prod.Link" /></td>

<td>@prod.Name</td>

<td>@prod.Description</td>

<td>@prod.Price</td>

<td>@prod.Quality</td>

<td>@(prod.Quality\*prod.Price)</td>

<td>

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

<input type="hidden" value="@prod.Id" name="id" />

<input type="submit" class="btn btn-danger" value="Удалить из корзины" />

</form>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

Home Create.cshtml

@model MobilePhoneBD.Models.ViewMode.CreateViewModel

@{

ViewData["Title"] = "Создание товара";

}

<form **asp-action**="Create" method="post" enctype="multipart/form-data">

<div class="card border shadow-lg p-0">

<div class="m-5">

<h3>Создание нового товара</h3>

<div class="">

<label>Название</label>

<p><input **asp-for**="Name" required /></p>

</div>

<div>

<label>Описание</label>

<p><input **asp-for**="Description" required /></p>

</div>

<div>

<label>Цена</label>

<p><input **asp-for**="Price" required **type**="number" /></p>

</div>

<div>

<label>Количество</label>

<p><input **asp-for**="Quality" required **type**="number" min="1" /></p>

</div>

<div>

<label>Авто</label>

<p>

<select **asp-for**="CatId" id="cat" onchange="GetMan()" required>

<option disabled selected>Выберите</option>

@foreach(var cat in Model.Category)

{

<option **value**="@cat.Id">@cat.Name</option>

}

</select>

</p>

</div>

<div>

<label>Запчасть</label>

<p>

<select **asp-for**="ManId" id="man" required>

</select>

</p>

</div>

<div>

<label>Фото</label>

<p><input **type**="file" **asp-for**="UploadedFile" required/></p>

</div>

<button type="submit" class="btn btn-success">Добавить</button>

</div>

</div>

</form>

Home Edit.cshtml

@model MobilePhoneBD.Models.ViewMode.EditViewModel

@{

ViewData["Title"] = "Редактирование товара";

}

<form **asp-action**="Edit" method="post" enctype="multipart/form-data">

<div class="card border shadow-lg p-0">

<div class="m-5">

<h3>Создание нового товара</h3>

<div class="">

<input **type**="hidden" **asp-for**="Id" **value**="@Model.Id" />

<label>Название</label>

<p><input **asp-for**="Name" required /></p>

</div>

<div>

<label>Описание</label>

<p><input **asp-for**="Description" required /></p>

</div>

<div>

<label>Цена</label>

<p><input **asp-for**="Price" required /></p>

</div>

<div>

<label>Количество</label>

<p><input **asp-for**="Quality" required **type**="number" min="1" /></p>

</div>

<div>

<label>Фото</label>

<p><input **type**="file" **asp-for**="File" /></p>

</div>

<button type="submit" class="btn btn-success">Изменить</button>

</div>

</div>

</form>

Home Index.cshtml

@model MobilePhoneBD.Models.ViewMode.IndexViewModel

@{ ViewData["Title"] = "Home Page"; }

<div class="text-center row justify-content-between">

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

<select name="catId" onchange="GetMan()" id="cat">

<option **value**="">Авто</option>

@foreach (var cat in Model.Category)

{

<option **value**="@cat.Id">@cat.Name</option>

}

</select>

<select name="manId" id="man" class="m-2">

<option **value**="">Деталь</option>

</select>

<label class="ml-2">Количество записей</label>

<select **asp-for**="Count" class="m-2">

<option **value**="0">Все</option>

<option **value**="5">5</option>

<option **value**="15">15</option>

<option **value**="30">30</option>

</select>

<button type="submit" class="btn btn-outline-info">Показать</button>

</form>

</div>

<div class="card shadow-lg p-3">

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

<h5 style="color:#4cff00">Создание Авто</h5>

<input **asp-for**="CreateViewModel.NewCat" />

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

</form>

</div>

<div class="card shadow-lg p-3 mt-3">

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

<h5 style="color:#0094ff">Создание запчасти</h5>

<select **asp-for**="CreateViewModel.CatId">

@foreach (var t in Model.Category)

{

<option **value**="@t.Id">@t.Name</option> }

</select>

<p>Название производителя</p>

<input **asp-for**="CreateViewModel.NewMan" />

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

</form>

</div>

<div>

@foreach (var prod in Model.Products)

{

<div class="card border shadow-lg m-2">

<table class="table">

<thead><tr><th>Фото</th><th>Название</th><th>Описание</th><th>Цена</th><th>Количество</th><th>Добавление</th><th></th><th></th></tr></thead>

<tbody>

<tr>

<td><img style="width:100px" src="@prod.Link" /></td>

<td>@prod.Name</td>

<td>@prod.Description</td>

<td>@prod.Price</td>

<td>@prod.Quality</td>

<td class="col-6">

<form **asp-action**="Add" **asp-controller**="Basket">

<input type="hidden" value="@prod.Id" name="id" />

<input type="number" class="form-control col-5 mt-1" max="@prod.Quality" min="1" name="quantity" />

<input type="submit" class="btn btn-success" value="Добавить" />

</form>

</td>

<td><button class="btn btn-info" type="submit" onclick="location.href ='/home/edit?id=@prod.Id'">Изменить</button></td>

<td>

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

<input type="hidden" value="@prod.Id" name="id" />

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

</form>

</td>

</tr>

</tbody>

</table>

</div>

}

</div>

Layout.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - AutoZap</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</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">

<a class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index" style="color:#0094ff">AutoZap</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-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 flex-sm-row-reverse">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Главная</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Basket" **asp-action**="Index">Корзина</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Create">Создать товар</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<div>

<div class="container" >

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2021 - AutoZap - <a **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</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>

@RenderSection("Scripts", required: false)

</body>

</html>

Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using MobilePhoneBD.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace MobilePhoneBD

{

public class Program

{

public static void Main(string[] args)

{

var host = CreateHostBuilder(args).Build();

using (var scope = host.Services.CreateScope())

{

var db = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();

db.Database.Migrate();

}

host.Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using MobilePhoneBD.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace MobilePhoneBD

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

// получаем строку подключения из файла конфигурации

string connection = Configuration.GetConnectionString("DefaultConnection");

// добавляем контекст MobileContext в качестве сервиса в приложение

services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connection));

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

//app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

Migrations

using Microsoft.EntityFrameworkCore.Migrations;

namespace AutoZap.Migrations

{

public partial class initial : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Auto",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: true),

Description = table.Column<string>(type: "nvarchar(max)", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Auto", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Zap",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: true),

AutoId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Zap", x => x.Id);

table.ForeignKey(

name: "FK\_Zap\_Auto\_AutoId",

column: x => x.AutoId,

principalTable: "Auto",

principalColumn: "Id",

onDelete: ReferentialAction.NoAction);

});

migrationBuilder.CreateTable(

name: "Products",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: true),

Quality = table.Column<int>(type: "int", nullable: false),

Description = table.Column<string>(type: "nvarchar(max)", nullable: true),

Price = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

Link = table.Column<string>(type: "nvarchar(max)", nullable: true),

AutoId = table.Column<int>(type: "int", nullable: false),

ZapId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Products", x => x.Id);

table.ForeignKey(

name: "FK\_Products\_Auto\_AutoId",

column: x => x.AutoId,

principalTable: "Auto",

principalColumn: "Id",

onDelete: ReferentialAction.NoAction);

table.ForeignKey(

name: "FK\_Products\_Zap\_ZapId",

column: x => x.ZapId,

principalTable: "Zap",

principalColumn: "Id",

onDelete: ReferentialAction.NoAction);

});

migrationBuilder.CreateTable(

name: "Baskets",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Quality = table.Column<int>(type: "int", nullable: false),

ProductId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Baskets", x => x.Id);

table.ForeignKey(

name: "FK\_Baskets\_Products\_ProductId",

column: x => x.ProductId,

principalTable: "Products",

principalColumn: "Id",

onDelete: ReferentialAction.NoAction);

});

migrationBuilder.CreateIndex(

name: "IX\_Baskets\_ProductId",

table: "Baskets",

column: "ProductId");

migrationBuilder.CreateIndex(

name: "IX\_Products\_AutoId",

table: "Products",

column: "AutoId");

migrationBuilder.CreateIndex(

name: "IX\_Products\_ZapId",

table: "Products",

column: "ZapId");

migrationBuilder.CreateIndex(

name: "IX\_Zap\_AutoId",

table: "Zap",

column: "AutoId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Baskets");

migrationBuilder.DropTable(

name: "Products");

migrationBuilder.DropTable(

name: "Zap");

migrationBuilder.DropTable(

name: "Auto");

}

}

}