**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

**по дисциплине: «Разработка приложений баз данных для информационных систем»**

на тему: Разработка моделей и контроллеров ASP.NET MVC приложения баз данных

Выполнил: студент гр. ИТП-31

Солодков М.А

Принял: доцент

Асенчик О.Д.

Гомель 2019

**Цель работы:** ознакомиться с возможностями ASP.NET Core MVC и Entity Framework Core для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание**

Создать с использованием ASP.NET Core MVC Web-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент middleware, контроллеры и представления для выборки и отображения информации из не менее чем 3 – таблиц базы данных с использованием механизма внедрение зависимостей.

Для этого необходимо создать:

* Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.
* Класс контекста данных.
* Другие классы, например, классы View Model и т.п. (при необходимости).
* Компонент middleware, вызываемый в классе Startup, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.
* Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.
* Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».
* Используя предварительно созданный и сконфигурированный в классе Startup профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута ResponseCache для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\*N+240 секунд, где N – номер вашего варианта.
* С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута ResponseCache.

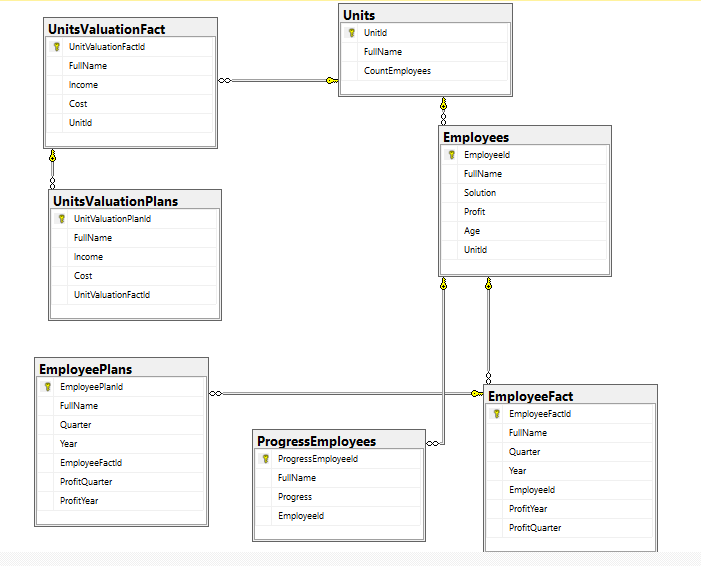


Рисунок 1 – Диаграмма базы данных

**Листинг программы Startup.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using CompanyASP.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using CompanyASP.Middleware;

using CompanyASP.Models;

using Microsoft.AspNetCore.Identity;

namespace Company

{

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)

{

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

services.AddDbContext<CompanyContext>(options => options.UseSqlServer(connection));

string connectionIdentity = Configuration.GetConnectionString("CompanyASPIdentityContextConnection");

services.AddDbContext<CompanyASPIdentityContext>(options => options.UseSqlServer(connectionIdentity));

services.AddIdentity<User, IdentityRole>()

.AddEntityFrameworkStores<CompanyASPIdentityContext>();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddResponseCaching();

services.AddMvc();

}

**Листинг программы DbInitializer.cs:**

using Microsoft.AspNetCore.Http;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CompanyASP.Data

{

public class DbInitializer

{

private readonly RequestDelegate next;

private static int employeeFactsCount = 100;

private static int employeePlansCount = 100;

private static int employeesCount = 100;

private static int progressEmployeeCount = 100;

private static int unitCount = 100;

private static int unitValuationFactsCount = 100;

private static int unitValuationPlansCount = 100;

private static Random randObj;

public DbInitializer(RequestDelegate next)

{

this.next = next;

}

public static void Initialize(CompanyContext db)

{

InitializeUnits(db);

InitializeEmployee(db);

InitializeUnitValuationFacts(db);

InitializeUnitValuationPlans(db);

InitializeEmployeeFacts(db);

InitializeEmployeePlans(db);

InitializeProgressEmployee(db);

}

private static void InitializeEmployeeFacts(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.EmployeeFacts.Any())

{

return;

}

else

{

string fullName;

int quarter;

int year;

decimal profitYear;

decimal profitQuarter;

int employeeId;

randObj = new Random(1);

string[] employeeFacts\_voc = { "Солодков М.А", "Трофимов Е.В", "Мартинович И.А", "Назаренко И.А", "Грамович А.В", "Брусенцова Ю.В", "Пархоменко П.И", "Колос В.В" };

int count\_employeeFacts\_voc = employeeFacts\_voc.GetLength(0);

for (int employeeFactId = 1; employeeFactId < employeeFactsCount; employeeFactId++)

{

fullName = employeeFacts\_voc[randObj.Next(count\_employeeFacts\_voc)] + employeeFactId.ToString();

quarter = (int)randObj.Next(30000, 50000);

year = (int)randObj.Next(60000, 100000);

profitQuarter = (decimal)randObj.Next(20000, 40000);

profitYear = (decimal)randObj.Next(50000, 90000);

employeeId = (int)randObj.Next(1, 99);

db.EmployeeFacts.Add(new Models.EmployeeFact { FullName = fullName, Quarter = quarter, Year = year, ProfitQuarter = profitQuarter, ProfitYear = profitYear, EmployeeID = employeeId });

}

db.SaveChanges();

}

}

private static void InitializeEmployeePlans(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.EmployeePlans.Any())

{

return;

}

else

{

string fullName;

int quarter;

int year;

decimal profitYear;

decimal profitQuarter;

int employeeFactId;

randObj = new Random(1);

string[] employeePlans\_voc = { "Солодков М.А", "Трофимов Е.В", "Мартинович И.А", "Назаренко И.А", "Грамович А.В", "Брусенцова Ю.В", "Пархоменко П.И", "Колос В.В" };

int count\_employeePlans\_voc = employeePlans\_voc.GetLength(0);

for (int employeePlanId = 1; employeePlanId < employeePlansCount; employeePlanId++)

{

fullName = employeePlans\_voc[randObj.Next(count\_employeePlans\_voc)] + employeePlanId.ToString();

quarter = (int)randObj.Next(50000, 70000);

year = (int)randObj.Next(80000, 120000);

profitQuarter = (decimal)randObj.Next(40000, 60000);

profitYear = (decimal)randObj.Next(70000, 110000);

employeeFactId = (int)randObj.Next(1, 99);

db.EmployeePlans.Add(new Models.EmployeePlan { FullName = fullName, Quarter = quarter, Year = year, ProfitQuarter = profitQuarter, ProfitYear = profitYear, EmployeeFactID = employeeFactId });

}

db.SaveChanges();

}

}

private static void InitializeEmployee(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.Employees.Any())

{

return;

}

else

{

string fullName;

decimal salary;

decimal profit;

int unitId;

int age;

randObj = new Random(1);

string[] employee\_voc = { "Солодков М.А", "Трофимов Е.В", "Мартинович И.А", "Назаренко И.А", "Грамович А.В", "Брусенцова Ю.В", "Пархоменко П.И", "Колос В.В" };

int count\_employee\_voc = employee\_voc.GetLength(0);

for (int employeeID = 1; employeeID < employeesCount; employeeID++)

{

fullName = employee\_voc[randObj.Next(count\_employee\_voc)] + employeeID.ToString();

salary = (decimal)randObj.Next(1000, 2000);

profit = (decimal)randObj.Next(1000, 1500);

age = (int)randObj.Next(20, 55);

unitId = (int)randObj.Next(1, 99);

db.Employees.Add(new Models.Employee { FullName = fullName, Salary = salary, Profit = profit, Age = age, UnitId = unitId });

}

db.SaveChanges();

}

}

private static void InitializeProgressEmployee(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.ProgressEmployees.Any())

{

return;

}

else

{

string fullName;

string progress;

int employeeId;

randObj = new Random(1);

string[] employee\_voc = { "Солодков М.А", "Трофимов Е.В", "Мартинович И.А", "Назаренко И.А", "Грамович А.В", "Брусенцова Ю.В", "Пархоменко П.И", "Колос В.В" };

string[] progress\_voc = { "выполнил работу вовремя", "просрочил дедлайн", "разлил кофе на клавиатуру", "подрался с коллегой за коржик", "получил прибавку к заралпте", "повысился в должности" };

int count\_employee\_voc = employee\_voc.GetLength(0);

int count\_progress\_voc = progress\_voc.GetLength(0);

for (int progressEmployeeId = 1; progressEmployeeId < progressEmployeeCount; progressEmployeeId++)

{

fullName = employee\_voc[randObj.Next(count\_employee\_voc)] + progressEmployeeId.ToString();

progress = progress\_voc[randObj.Next(count\_progress\_voc)];

employeeId = (int)randObj.Next(1, 99);

db.ProgressEmployees.Add(new Models.ProgressEmployee { FullName = fullName, Progress = progress, EmployeeID = employeeId });

}

db.SaveChanges();

}

}

private static void InitializeUnits(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.Units.Any())

{

return;

}

else

{

string unitName;

int countEmployee;

Random randObj = new Random(1);

string[] units\_voc = { "администрация", "отдел кадров", "отдел маркетинга", "отдел продаж", "отдел финансов", "отдел логистики", "отдел IT", "отдел закупок", "отдел исследования и развития" };

int count\_units\_voc = units\_voc.GetLength(0);

for (int unitId = 1; unitId < unitCount; unitId++)

{

unitName = units\_voc[randObj.Next(count\_units\_voc)] + unitId.ToString();

countEmployee = (int)randObj.Next(50, 150);

db.Units.Add(new Models.Unit {FullName = unitName, CountEmployee = countEmployee });

}

db.SaveChanges();

}

}

private static void InitializeUnitValuationFacts(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.UnitValuationFacts.Any())

{

return;

}

else

{

string unitName;

decimal profit;

decimal salary;

int unitId;

randObj = new Random(1);

string[] units\_voc = { "администрация", "отдел кадров", "отдел маркетинга", "отдел продаж", "отдел финансов", "отдел логистики", "отдел IT", "отдел закупок", "отдел исследования и развития" };

int count\_units\_voc = units\_voc.GetLength(0);

for(var unitValuationFactId = 1; unitValuationFactId < unitValuationFactsCount; unitValuationFactId++)

{

unitName = units\_voc[randObj.Next(count\_units\_voc)] + unitValuationFactId.ToString();

profit = (decimal)randObj.Next(700000, 1000000);

salary = (decimal)randObj.Next(850000, 1200000);

unitId = (int)randObj.Next(1, 99);

db.UnitValuationFacts.Add(new Models.UnitValuationFact { FullName = unitName, Profit = profit, Salary = salary, UnitID = unitId });

}

db.SaveChanges();

}

}

private static void InitializeUnitValuationPlans(CompanyContext db)

{

db.Database.EnsureCreated();

if (db.UnitValuationPlans.Any())

{

return;

}

else

{

string unitName;

decimal profit;

decimal salary;

int unitValuationFactId;

randObj = new Random(1);

string[] units\_voc = { "администрация", "отдел кадров", "отдел маркетинга", "отдел продаж", "отдел финансов", "отдел логистики", "отдел IT", "отдел закупок", "отдел исследования и развития" };

int count\_units\_voc = units\_voc.GetLength(0);

for (var unitValuationPlanId = 1; unitValuationPlanId < unitValuationPlansCount; unitValuationPlanId++)

{

unitName = units\_voc[randObj.Next(count\_units\_voc)] + unitValuationPlanId.ToString();

profit = (decimal)randObj.Next(500000, 700000);

salary = (decimal)randObj.Next(950000, 1350000);

unitValuationFactId = (int)randObj.Next(1, 99);

db.UnitValuationPlans.Add(new Models.UnitValuationPlan { FullName = unitName, Profit = profit, Salary = salary, UnitValuationFactID = unitValuationFactId});

}

db.SaveChanges();

}

}

}

}

**Листинг программы EmployeesController.cs:**

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 CompanyASP.Data;

using CompanyASP.Models;

namespace CompanyASP.Controllers

{

public class EmployeesController : Controller

{

private readonly CompanyContext \_context;

public EmployeesController(CompanyContext context)

{

\_context = context;

}

// GET: Employees

[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 270)]

public async Task<IActionResult> Index()

{

var companyContext = \_context.Employees.Include(e => e.Unit);

return View(await companyContext.ToListAsync());

}

// GET: Employees/Details/5

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

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.Unit)

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

if (employee == null)

{

return NotFound();

}

return View(employee);

}

// GET: Employees/Create

public IActionResult Create()

{

ViewData["UnitId"] = new SelectList(\_context.Units, "UnitID", "UnitID");

return View();

}

// POST: Employees/Create

// To protect from overposting attacks, please 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("EmployeeID,FullName,Salary,Profit,Age,UnitId")] Employee employee)

{

if (ModelState.IsValid)

{

\_context.Add(employee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["UnitId"] = new SelectList(\_context.Units, "UnitID", "UnitID", employee.UnitId);

return View(employee);

}

// GET: Employees/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

ViewData["UnitId"] = new SelectList(\_context.Units, "UnitID", "UnitID", employee.UnitId);

return View(employee);

}

// POST: Employees/Edit/5

// To protect from overposting attacks, please 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("EmployeeID,FullName,Salary,Profit,Age,UnitId")] Employee employee)

{

if (id != employee.EmployeeID)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(employee);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(employee.EmployeeID))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["UnitId"] = new SelectList(\_context.Units, "UnitID", "UnitID", employee.UnitId);

return View(employee);

}

// GET: Employees/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.Unit)

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

if (employee == null)

{

return NotFound();

}

return View(employee);

}

// POST: Employees/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var employee = await \_context.Employees.FindAsync(id);

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool EmployeeExists(int id)

{

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

}

}

}

**Листинг программы ProgressEmployeeController.cs:**

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 CompanyASP.Data;

using CompanyASP.Models;

namespace CompanyASP.Controllers

{

public class ProgressEmployeesController : Controller

{

private readonly CompanyContext \_context;

public ProgressEmployeesController(CompanyContext context)

{

\_context = context;

}

// GET: ProgressEmployees

[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 270)]

public async Task<IActionResult> Index()

{

var companyContext = \_context.ProgressEmployees.Include(p => p.Employee);

return View(await companyContext.ToListAsync());

}

// GET: ProgressEmployees/Details/5

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

{

if (id == null)

{

return NotFound();

}

var progressEmployee = await \_context.ProgressEmployees

.Include(p => p.Employee)

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

if (progressEmployee == null)

{

return NotFound();

}

return View(progressEmployee);

}

// GET: ProgressEmployees/Create

public IActionResult Create()

{

ViewData["EmployeeID"] = new SelectList(\_context.Employees, "EmployeeID", "EmployeeID");

return View();

}

// POST: ProgressEmployees/Create

// To protect from overposting attacks, please 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("ProgressEmployeeID,FullName,Progress,EmployeeID")] ProgressEmployee progressEmployee)

{

if (ModelState.IsValid)

{

\_context.Add(progressEmployee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["EmployeeID"] = new SelectList(\_context.Employees, "EmployeeID", "EmployeeID", progressEmployee.EmployeeID);

return View(progressEmployee);

}

// GET: ProgressEmployees/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var progressEmployee = await \_context.ProgressEmployees.FindAsync(id);

if (progressEmployee == null)

{

return NotFound();

}

ViewData["EmployeeID"] = new SelectList(\_context.Employees, "EmployeeID", "EmployeeID", progressEmployee.EmployeeID);

return View(progressEmployee);

}

// POST: ProgressEmployees/Edit/5

// To protect from overposting attacks, please 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("ProgressEmployeeID,FullName,Progress,EmployeeID")] ProgressEmployee progressEmployee)

{

if (id != progressEmployee.ProgressEmployeeID)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(progressEmployee);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ProgressEmployeeExists(progressEmployee.ProgressEmployeeID))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["EmployeeID"] = new SelectList(\_context.Employees, "EmployeeID", "EmployeeID", progressEmployee.EmployeeID);

return View(progressEmployee);

}

// GET: ProgressEmployees/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var progressEmployee = await \_context.ProgressEmployees

.Include(p => p.Employee)

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

if (progressEmployee == null)

{

return NotFound();

}

return View(progressEmployee);

}

// POST: ProgressEmployees/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var progressEmployee = await \_context.ProgressEmployees.FindAsync(id);

\_context.ProgressEmployees.Remove(progressEmployee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool ProgressEmployeeExists(int id)

{

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

}

}

}

**Листинг программы UnitsController.cs:**

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 CompanyASP.Data;

using CompanyASP.Models;

namespace CompanyASP.Controllers

{

public class UnitsController : Controller

{

private readonly CompanyContext \_context;

public UnitsController(CompanyContext context)

{

\_context = context;

}

// GET: Units

[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 270)]

public async Task<IActionResult> Index()

{

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

}

// GET: Units/Details/5

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

{

if (id == null)

{

return NotFound();

}

var unit = await \_context.Units

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

if (unit == null)

{

return NotFound();

}

return View(unit);

}

// GET: Units/Create

public IActionResult Create()

{

return View();

}

// POST: Units/Create

// To protect from overposting attacks, please 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("UnitID,FullName,CountEmployee")] Unit unit)

{

if (ModelState.IsValid)

{

\_context.Add(unit);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(unit);

}

// GET: Units/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var unit = await \_context.Units.FindAsync(id);

if (unit == null)

{

return NotFound();

}

return View(unit);

}

// POST: Units/Edit/5

// To protect from overposting attacks, please 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("UnitID,FullName,CountEmployee")] Unit unit)

{

if (id != unit.UnitID)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(unit);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UnitExists(unit.UnitID))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(unit);

}

// GET: Units/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var unit = await \_context.Units

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

if (unit == null)

{

return NotFound();

}

return View(unit);

}

// POST: Units/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var unit = await \_context.Units.FindAsync(id);

\_context.Units.Remove(unit);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool UnitExists(int id)

{

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

}

}

}

**Результат выполнения:**

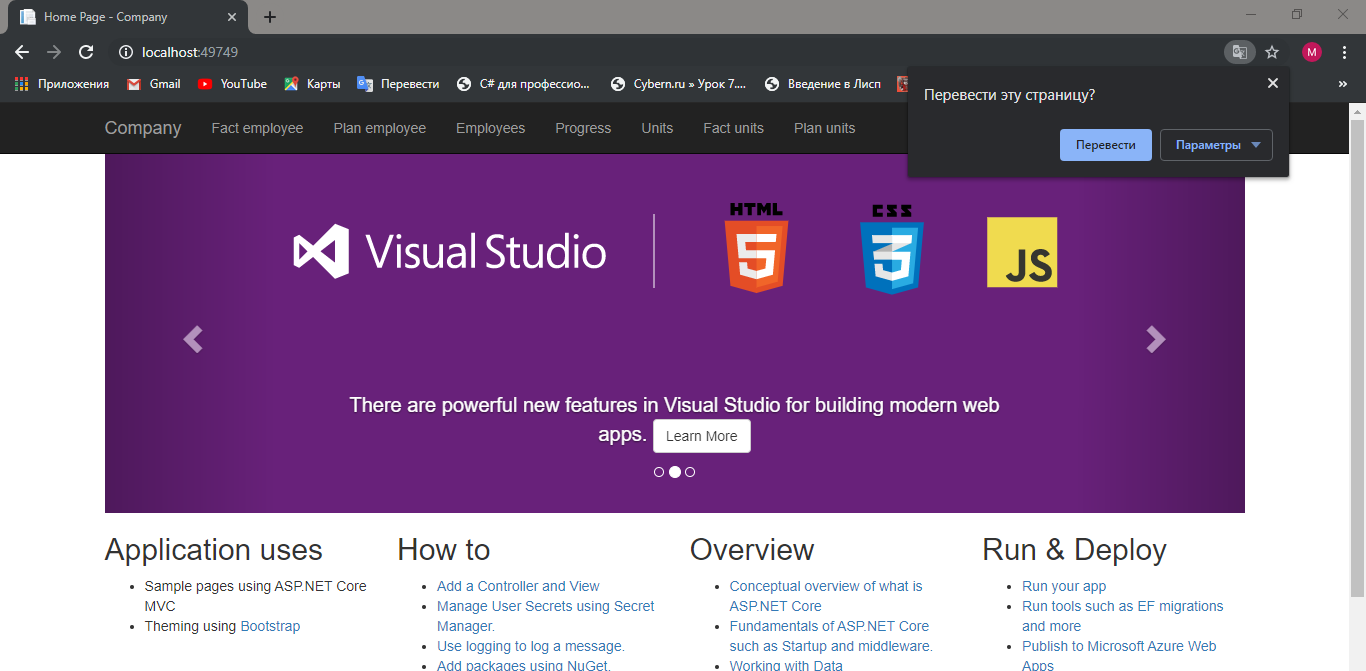


Рисунок 2 – Главная страница сайта

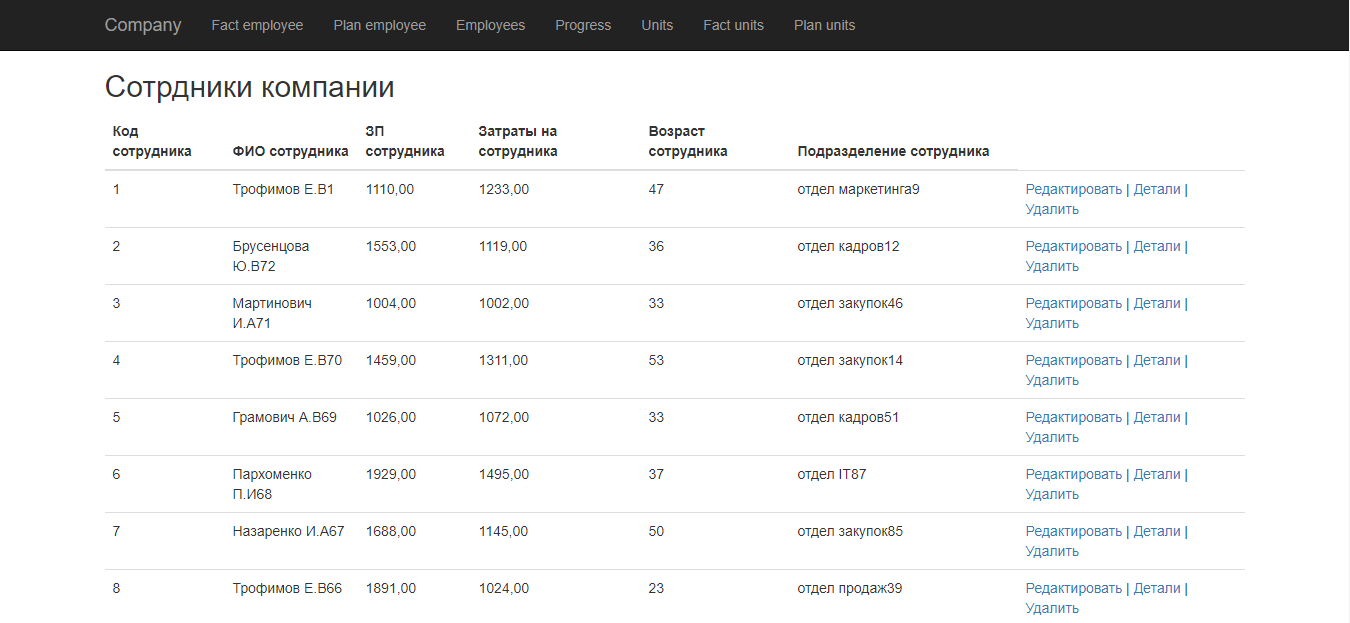


Рисунок 3 – Выборка из таблицы Employee

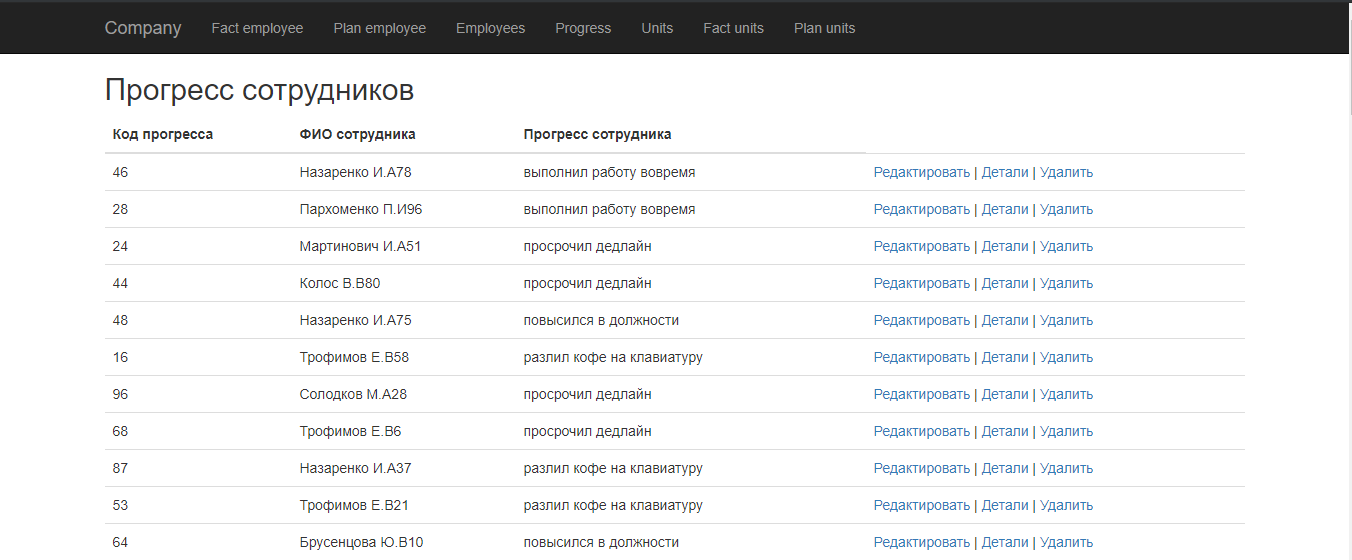


Рисунок 4 – Выборка из таблицы ProgressEmployee

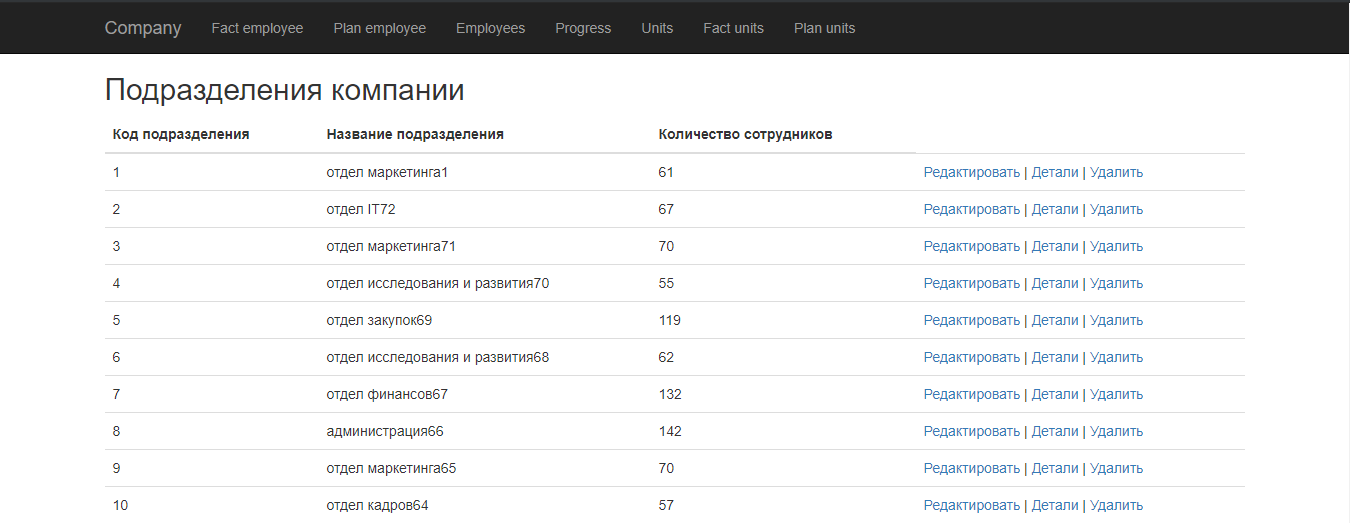


Рисунок 5 – Выборка из таблицы Units

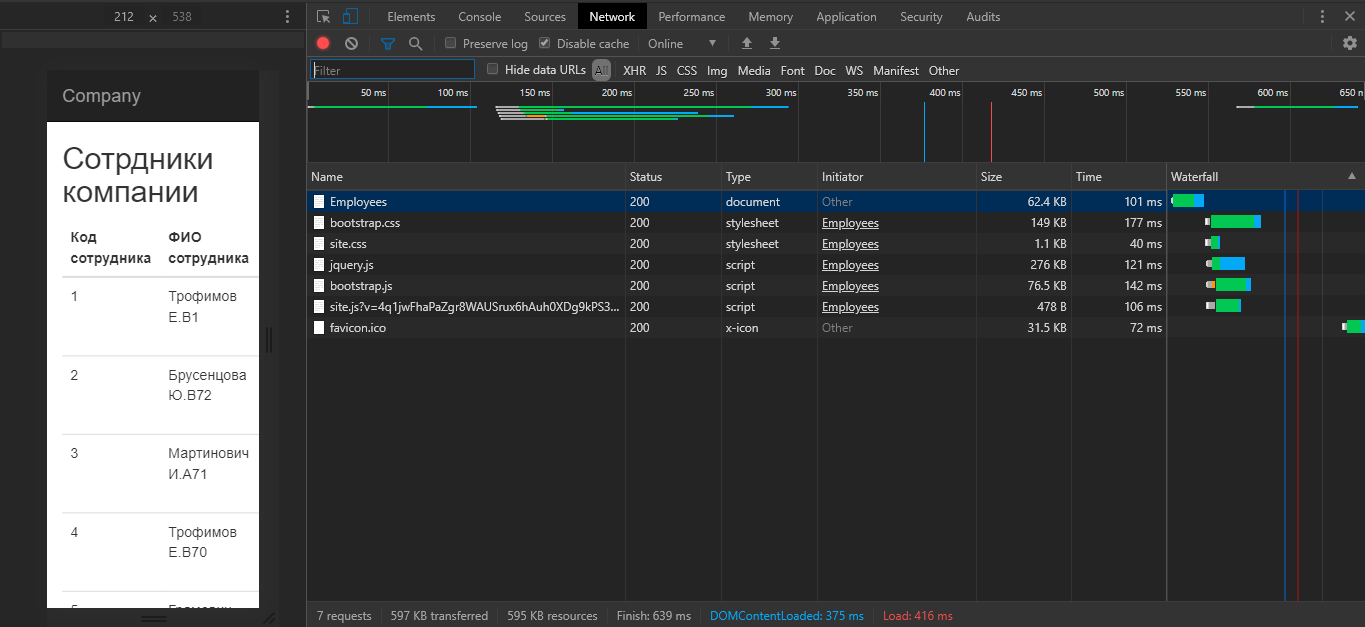


Рисунок 6 – Обработка запроса из таблицы Employee при отсутствии кэширования с использованием ResponseCache

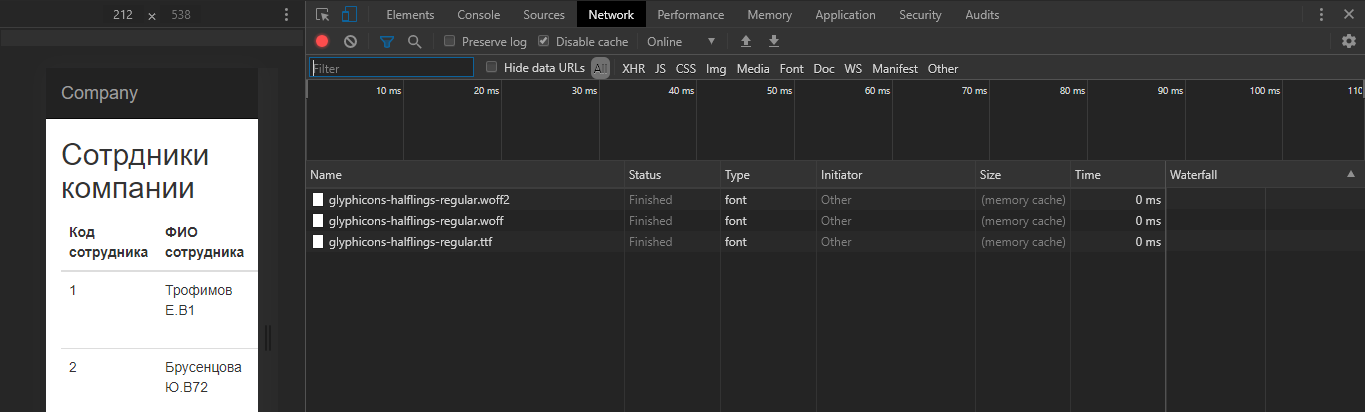


Рисунок 7 – Продемонстрировать ускорение обработки запроса при наличии кэширования с использованием ResponseCache

**Вывод:** в ходе лабораторной работы ознакомились с возможностями ASP.NET Core MVC и Entity Framework Core для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.