МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

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

на тему: «Обработка *HTTP* запросов средствами *ASP.NET Core*. Сохранение состояния. Кэширование.»

Выполнил: студент гр. ИТП-31
Король В. Н.
Принял: ректор
Асенчик О.Д.

Гомель 2023

Цель работы: ознакомиться с методами обработкой *HTTP* средствами *ASP.NET Core*, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

Задание:

Используя ранее разработанные объектную модель для доступа к данным в заданной предметной области разработать простое *ASP.NET Core* приложение.

2.1. С использование методов *Run*, *Map* и *Use* разработать:

1. компоненты промежуточного уровня (middleware) и встроить их в конвейер обработки *HTTP* запроса с целью кэширования 20 записей из каждой таблицы базы данных заданной предметной области с помощью встроенного инструмента кэширования – объекта IMemoryCache. Данные в кэше хранить неизменными в течение $2*N+240$ секунд, где N – номер вашего варианта.

2. собственную систему маршрутизации входящих запросов:

• если *URL* адрес входящего запроса содержит *\info* – выводить в выходной поток для отображения браузером информацию о клиенте и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит *\table* (где *table* – имя таблицы из базы данных) – выводить в выходной поток для отображения браузером с использование метода *Response.WriteAsync* кэшированную информацию из соответствующей таблицы базы данных и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит *\searchform1* или *\searchform2* – выводить в выходной поток для отображения браузером с использование метода *Response.WriteAsync* формы для поиска информации из базы данных и выходить из конвейера обработки запроса;

форма должна содержать, как минимум: одно поле, одного поле со списком, один список, одну кнопку;

• в противном случае (*URL* адрес входящего запроса не содержит перечисленных выше элементов) – продолжать обрабатывать другие компоненты конвейера обработки запросов;

2.2. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки (*\searchform1*).

2.3. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта *Session* (*\searchform2*).

2.4. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте *Session* и куки (*\searchform1*, *\searchform2*).

2.5. С использованием средств разработчика браузера (*Chrome, Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием *MemoryCache*.

2.6. Разместить выполненный проект на *github*.

Ход работы

В первую очередь при выполнении лабораторной работы база данных созданная в первой лабораторной работе была перенесена в проект при помощи технологии *ENTITY FRAMEWORK*. После перенесения базы в проект были сгенерированы классы моделей и класс контекста. Далее после подключения *Entity Framework* строка подключения была перенесена в конфигурационный файл *appsetings.json* далее при помощи класса *WebApplicationBuilder* файл контекста был внедрен в при помощи *DI* в главный класс проекта. Теперь для обращения к базе данных будет использоваться один объект класса контекста. Листинг этих классов указан в приложении А.

Далее после подключения базы данных к проекту был создан класс *InsuranceCompanyCache* который используется для кэширования запросов к базе данных. Для этого был создан метод *GetEntities* который принимает входные параметры название таблицы, которую надо получить из кэша и количество строк, которые надо получить. Для добавления кэширования каждой таблице необходимо будет реализовать метод для каждой модели чтобы избежать этого был реализован интерфейс *IEntity* который наследует каждый класс модели. После этого был реализован только один класс для получения всех моделей из кэша. Для получения необходимой модели по ее названии был создан класс *InsuranceCompanyFactory* который реализует паттерн фабричный метод. Листинг этих классов указан в приложении А.

Далее после создания класса для кэширования данных был реализован класс *InsuranceCompanyHandlers* который используется для обработки запросов по определенному *URL* адресу. Листинг этих классов указан в приложении А.

Для генерации *HTML* страниц был реализован класс *HtmlBuilder* который реализует паттерн строитель и позволяет поэтапно генерировать *html* страницу. Для более удобной генерации *html* страниц был реализован интерфейс *IHtmlVisitor* от которого наследуются классы *HtmlTableVisitor* для отрисовки таблиц каждой модели и *HtmlFormVisitor* для отрисовки формы каждой модели. Листинг этих классов указан в приложении А.

После реализации *html* генераторов были реализованы классы *CookiesVisitor* и *SessionsVisitor* которые используются для сохранения данных в сессии и куки. Листинг этих классов указан в приложении А.

Далее после создания всех классов были установлены все связи между *url* адресами и обработчиками. Пример страницы с главным меню приложения указаны на рисунке 1.

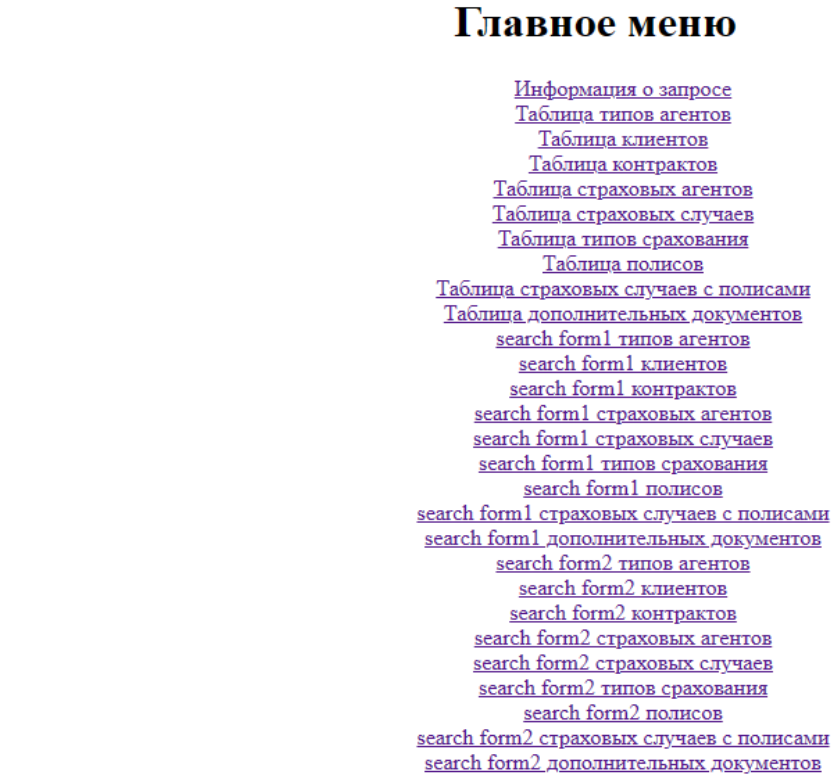

Рисунок 1 – Пример главного меню приложения

Далее была разработана страница для вывода информации о запросе пользователя. Пример этой страницы указан на рисунке 2.

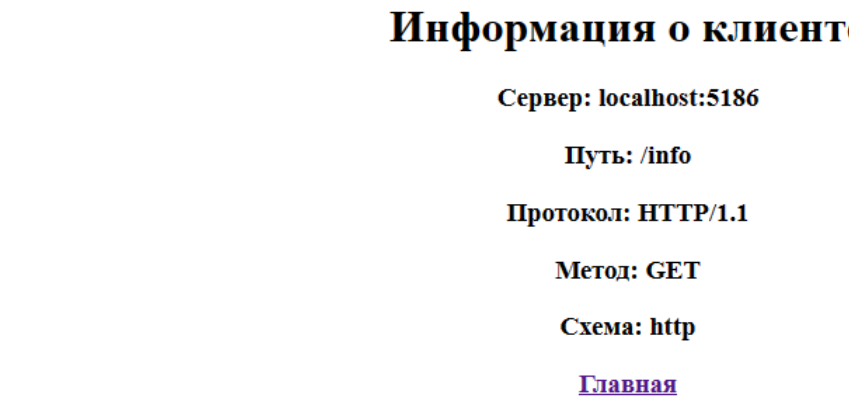

Рисунок 2 – Пример страницы с информацией о запросе клиента

Далее были реализованы страницы, которые выводят данные из базы в виде таблицы. Для ускорения обработки запросов были реализованы методы кэширования. Для сравнения ускорения запросов с кэшированием и без была замерена скорость запроса без кэширования, а потом с ним. Пример скорости запроса без кэширования указан на рисунке 3.

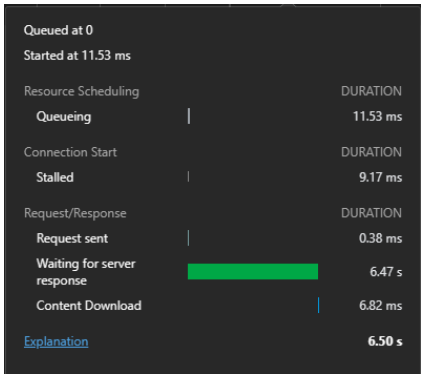

Рисунок 3 – Скорость выполнения запроса без кэширования

Далее была замерена скорость обработки запроса с кэшированием. Пример скорости запроса с кэширования указан на рисунке 4.

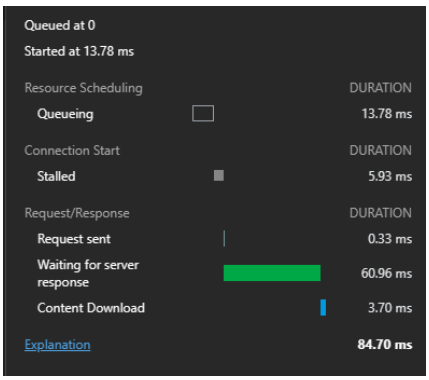

Рисунок 4 – Скорость выполнения запроса с кэширования

Далее после получения данных из кэша они передаются в метод для отрисовки таблицы на основе модели. Пример таблицы клиентов указана на рисунке 5.

1	Елена	Александров	Михайлович	12/21/2018 12:00:00 AM	+375410354177	Каир	Холодильная улица	9	5	TU5052472	1377363U735AK4	4/7/2015 12:00:00 AM
2	Дмитрий	Иванов	Дмитриевич	5/19/2027 12:00:00 AM	+375823410411	Рим	Гусарская улица	80	5	NO8060637	3324953Q497WX9	4/22/2026 12:00:00 AM
3	Михаил	Кузнецов	Сидорович	7/25/2031 12:00:00 AM	+375172691090	Москва	Толстого улица	86	11	AC4104051	1654074B168IC6	12/14/2031 12:00:00 AM
4	Елена	Сидоров	Сергеевич	11/28/2031 12:00:00 AM	+375913743850	Санкт-Петербург	Лесная улица	25	6	LD8426389	6671186K109UI3	2/1/2033 12:00:00 AM
5	Мария	Иванов	Сергеевич	12/2/2026 12:00:00 AM	+375476259409	Рио-де-Жанейро	Шевченко улица	38	20	KO7723115	6531521J887SM9	10/22/2019 12:00:00 AM
6	Елена	Иванов	Михайлович	3/6/2014 12:00:00 AM	+375647550006	Каир	Театральная улица	54	2	ES2534657	1084396V502JY9	1/19/2033 12:00:00 AM
7	Петр	Александров	Дмитриевич	9/25/2030 12:00:00 AM	+375729477121	Санкт-Петербург	Фрунзе улица	16	9	DQ1523729	2592946K762XJ0	6/23/2024 12:00:00 AM
8	Мария	Иванов	Михайлович	9/3/2031 12:00:00 AM	+375126319842	Лос-Анджелес	Строителей улица	35	14	OC1888402	8076808C197BA1	8/2/2031 12:00:00 AM
9	София	Иванов	Андреевич	2/25/2019 12:00:00 AM	+375444906776	Дублин	Сельская улица	66	1	KR1311613	9997436J737WG0	1/30/2015 12:00:00 AM
10	Мария	Петров	Андреевич	11/24/2031 12:00:00 AM	+375685295294	Лондон	Свердловская улица	11	12	YR1438273	0663196B526FV5	1/15/2024 12:00:00 AM
11	Александр	Сидоров	Николаевич	7/21/2014 12:00:00 AM	+375100464546	Рим	Лесная улица	50	15	GH4238625	8538485J677WT8	10/16/2017 12:00:00 AM
12	Мария	Александров	Николаевич	12/2/2022 12:00:00 AM	+375826311816	Амстердам	Московская улица	65	3	CZ2808364	9245791N847SH5	11/16/2019 12:00:00 AM
13	Екатерина	Егоров	Петрович	7/5/2018 12:00:00 AM	+375854537453	Токио	Сельская улица	69	6	EO9319221	6424131W894NH1	12/19/2015 12:00:00 AM
14	Петр	Иванов	Петрович	2/8/2016 12:00:00 AM	+375519729137	Мумбаи	Полярная улица	99	9	GE9946338	2018917K413PN6	3/17/2030 12:00:00 AM
15	София	Смирнов	Николаевич	10/24/2013 12:00:00 AM	+375503263559	Мумбаи	Парковая улица	6	20	TK8504122	8180867P062JF1	11/26/2017 12:00:00 AM
16	Иван	Васильев	Дмитриевич	4/28/2023 12:00:00 AM	+375071753563	Рио-де-Жанейро	Парковая улица	38	6	JI1062087	1562049Q825FQ6	2/25/2033 12:00:00 AM
17	Александр	Козлов	Геннадьевич	9/3/2032 12:00:00 AM	+375954896517	Париж	Сосновая улица	20	9	SG0120648	0417663H732KU3	1/29/2023 12:00:00 AM
18	Мария	Егоров	Геннадьевич	10/17/2028 12:00:00 AM	+375089143009	Дублин	Смирнова улица	63	18	YH9739149	3500659Y887GL9	6/4/2032 12:00:00 AM
19	Петр	Иванов	Михайлович	11/25/2020 12:00:00 AM	+375729468532	Мадрид	Гранитная улица	70	18	LG9758660	7624390M390KZ0	1/27/2018 12:00:00 AM
20	Иван	Козлов	Сергеевич	1/22/2015 12:00:00 AM	+375822361180	Дублин	Молодежная улица	67	5	SU3774859	0366023W525IX6	9/1/2017 12:00:00 AM

Главная

Рисунок 5 – Пример таблицы клиентов

Далее были реализованы страницы, которые реализуют поля для фильтрации данных в таблице. Поля, которые используются для ввода данных для фильтрации сохраняют данные в *Sessions*. Пример страницы с поиском указан на рисунке 6.

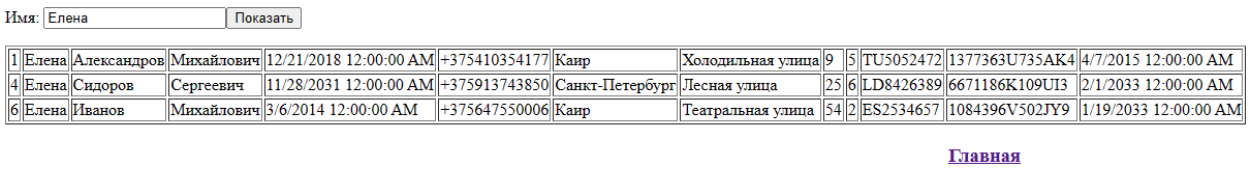

Рисунок 6 – Пример таблицы клиентов с формой через *Sessions* для фильтрации

Далее были реализованы страницы, которые реализуют поля для фильтрации данных в таблице. Поля, которые используются для ввода данных для фильтрации сохраняют данные в *Cookies*. Пример страницы с поиском указан на рисунке 7.

Рисунок 7 – Пример таблицы клиентов с формой через *Cookies* для фильтрации

Пример хранения данных в *Session* и *Cookies* указаны на рисунке 8.

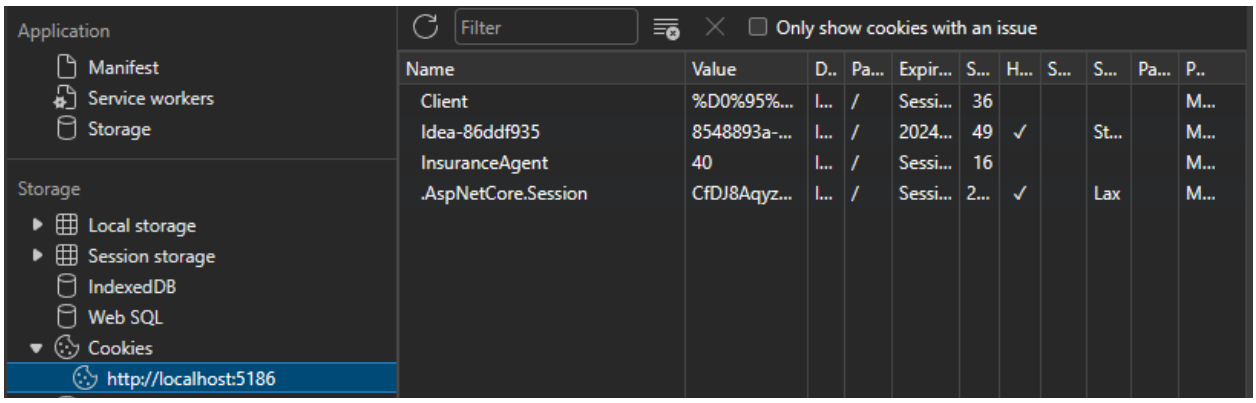

Рисунок 8 – Пример хранения данных в *Session* и *Cookies*

После выполнения лабораторной работы созданные проект был добавлен в локальный *git* репозиторий а потом перенесен в *GitHub* репозиторий своего аккаунта. Чтобы ознакомится с созданным проектом можно по ссылке *Javaro3/lab3_DB(github.com)*.

Вывод: в ходе выполнения лабораторной работы была изучена такая технология *ASP .NET* для создания веб приложений. Была изучена технология кэширования данных при помощи интерфейса *IMemoryCache*. Были изучены механизмы обработки запросов при помощи класса *HttpContext*. Были изучены методы сохранения данных во временное хранилище при помощи технологии *Sessions* и *Cookies*.

ПРИЛОЖНИЕ А

Листинг класса *InsuranceCompanyContext*

```
using lab3.Models;
using Microsoft.EntityFrameworkCore;

namespace lab3.Data;

public partial class InsuranceCompanyContext : DbContext
{
    public static List<string> DbSetNames = new InsuranceCompanyContext()
        .GetType()
        .GetProperties()
        .Where(p => p.PropertyType.IsGenericType &&
p.PropertyType.GetGenericTypeDefinition() == typeof(DbSet<>))
        .Select(p => p.Name)
        .ToList();

    public InsuranceCompanyContext()
    {
    }

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

    public virtual DbSet<AgentType> AgentTypes { get; set; }

    public virtual DbSet<Client> Clients { get; set; }

    public virtual DbSet<Contract> Contracts { get; set; }

    public virtual DbSet<InsuranceAgent> InsuranceAgents { get; set; }

    public virtual DbSet<InsuranceCase> InsuranceCases { get; set; }

    public virtual DbSet<InsuranceType> InsuranceTypes { get; set; }

    public virtual DbSet<Policy> Policies { get; set; }

    public virtual DbSet<PolicyInsuranceCase> PolicyInsuranceCases { get; set; }

    public virtual DbSet<SupportingDocument> SupportingDocuments { get; set; }


    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<AgentType>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__AgentTyp__3214EC075CDF68EC");

            entity.Property(e => e.Type)
                .HasMaxLength(50)
                .IsFixedLength();
        });

        modelBuilder.Entity<Client>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Clients__3214EC0743EDA985");

            entity.Property(e => e.Apartment).HasMaxLength(50);
            entity.Property(e => e.Birthdate).HasColumnType("date");
            entity.Property(e => e.City).HasMaxLength(50);
```

```
            entity.Property(e => e.House).HasMaxLength(50);
            entity.Property(e => e.MiddleName).HasMaxLength(20);
            entity.Property(e => e.MobilePhone).HasMaxLength(20);
            entity.Property(e => e.Name).HasMaxLength(20);
            entity.Property(e => e.PassportIdentification)
                .HasMaxLength(14)
                .IsFixedLength();
            entity.Property(e => e.PassportIssueDate).HasColumnType("date");
            entity.Property(e => e.PassportNumber)
                .HasMaxLength(9)
                .IsFixedLength();
            entity.Property(e => e.Street).HasMaxLength(50);
            entity.Property(e => e.Surname).HasMaxLength(20);
        });

        modelBuilder.Entity<Contract>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Contract__3214EC071E24DAA3");

            entity.Property(e => e.EndDeadline).HasColumnType("date");
            entity.Property(e => e.Responsibilities).HasMaxLength(100);
            entity.Property(e => e.StartDeadline).HasColumnType("date");
        });

        modelBuilder.Entity<InsuranceAgent>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Insuranc__3214EC071B7233A7");

            entity.Property(e => e.MiddleName).HasMaxLength(20);
            entity.Property(e => e.Name).HasMaxLength(20);
            entity.Property(e => e.Salary).HasColumnType("money");
            entity.Property(e => e.Surname).HasMaxLength(20);

            entity.HasOne(d => d.AgentTypeNavigation).WithMany(p =>
p.InsuranceAgents)
                .HasForeignKey(d => d.AgentType)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_InsuranceAgents_AgentTypes");

            entity.HasOne(d => d.ContractNavigation).WithMany(p =>
p.InsuranceAgents)
                .HasForeignKey(d => d.Contract)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_InsuranceAgents_Contracts");
        });

        modelBuilder.Entity<InsuranceCase>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Insuranc__3214EC07245436C3");

            entity.Property(e => e.Date).HasColumnType("date");
            entity.Property(e => e.Description).HasColumnType("text");
            entity.Property(e => e.InsurancePayment).HasColumnType("money");

            entity.HasOne(d => d.ClientNavigation).WithMany(p => p.InsuranceCases)
                .HasForeignKey(d => d.Client)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_InsuranceCases_Clients");

            entity.HasOne(d => d.InsuranceAgentNavigation).WithMany(p =>
p.InsuranceCases)
                .HasForeignKey(d => d.InsuranceAgent)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_InsuranceCases_InsuranceAgents");
```

```
            entity.HasOne(d => d.SupportingDocumentNavigation).WithMany(p =>
p.InsuranceCases)
                .HasForeignKey(d => d.SupportingDocument)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_InsuranceCases_SupportingDocuments");
        });

        modelBuilder.Entity<InsuranceType>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Insuranc__3214EC0795A5D734");

            entity.Property(e => e.Description).HasColumnType("text");
            entity.Property(e => e.Name).HasMaxLength(100);
        });

        modelBuilder.Entity<Policy>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Policies__3214EC0721D8E412");

            entity.Property(e => e.ApplicationDate).HasColumnType("date");
            entity.Property(e => e.PolicyNumber)
                .HasMaxLength(16)
                .IsFixedLength();
            entity.Property(e => e.PolicyPayment).HasColumnType("money");

            entity.HasOne(d => d.ClientNavigation).WithMany(p => p.Policies)
                .HasForeignKey(d => d.Client)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_Policies_Clients");

            entity.HasOne(d => d.InsuranceAgentNavigation).WithMany(p => p.Policies)
                .HasForeignKey(d => d.InsuranceAgent)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_Policies_InsuranceAgents");

            entity.HasOne(d => d.InsuranceTypeNavigation).WithMany(p => p.Policies)
                .HasForeignKey(d => d.InsuranceType)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_Policies_InsuranceTypes");
        });

        modelBuilder.Entity<PolicyInsuranceCase>(entity =>
        {
            entity.HasNoKey();

            entity.HasOne(d => d.InsuranceCase).WithMany()
                .HasForeignKey(d => d.InsuranceCaseId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_PolicyInsuranceCases_InsuranceCases");

            entity.HasOne(d => d.Policy).WithMany()
                .HasForeignKey(d => d.PolicyId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("FK_PolicyInsuranceCases_Policies");
        });

        modelBuilder.Entity<SupportingDocument>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__Supporti__3214EC07C5B6BCD5");

            entity.Property(e => e.Description).HasColumnType("text");
            entity.Property(e => e.Name).HasMaxLength(100);
        });

        OnModelCreatingPartial(modelBuilder);
```

```
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

Листинг класса *InsuranceCompanyFactory*

```
using lab3.Models;

namespace lab3.Data {
    public static class InsuranceCompanyFactory {
        public static IEnumerable<IEntity> GetEnites(string entityName,
InsuranceCompanyContext db) {
            switch (entityName) {
                case "AgentTypes":
                    return db.AgentTypes;
                case "Clients":
                    return db.Clients;
                case "Contracts":
                    return db.Contracts;
                case "InsuranceAgents":
                    return db.InsuranceAgents;
                case "InsuranceCases":
                    return db.InsuranceCases;
                case "InsuranceTypes":
                    return db.InsuranceTypes;
                case "Policies":
                    return db.Policies;
                case "PolicyInsuranceCases":
                    return db.PolicyInsuranceCases;
                case "SupportingDocuments":
                    return db.SupportingDocuments;
            }
            return null;
        }
    }
}
```

Листинг класса *InsuranceCompanyCache*

```
using lab3.Models;
using Microsoft.Extensions.Caching.Memory;

namespace lab3.Data
{
    public class InsuranceCompanyCache
    {
        private IMemoryCache _cache;
        private InsuranceCompanyContext _db;
        private const int CACHE_SAVE_TIME = 2 * 16 + 240;


        public InsuranceCompanyCache(InsuranceCompanyContext db, IMemoryCache
memoryCache)
        {
            _db = db;
            _cache = memoryCache;
        }

        public IEnumerable<IEntity> GetEnites(string entityName, int rowCount = 20)
        {
            _cache.TryGetValue(entityName, out IEnumerable<IEntity>? entities);

            if (entities is null)
            {
```

```
                entities = InsuranceCompanyFactory.GetEnites(entityName,
_db).Take(rowCount);
                _cache.Set($"{entityName}{rowCount}", entities, new
MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(CACHE_SAVE_TIME
)));
            }
            return entities;
        }
    }
}
```

Листинг класса *CacheFilter*

```
using lab3.Models;
using Microsoft.IdentityModel.Tokens;

namespace lab3.Services
{
    public class CacheFilter : IFilterVisitor {
        public IEnumerable<AgentType> Filter(IEnumerable<AgentType> agentTypes,
AgentType agentType) {
            return agentTypes.Where(e => agentType.Type.IsNullOrEmpty() ||
e.Type.Trim() == agentType.Type.Trim());
        }

        public IEnumerable<Client> Filter(IEnumerable<Client> clients, Client
client) {
            return clients.Where(e => client.Name.IsNullOrEmpty() || e.Name.Trim()
== client.Name.Trim());
        }

        public IEnumerable<Contract> Filter(IEnumerable<Contract> contracts,
Contract contract) {
            return contracts.Where(e => contract.Responsibilities.IsNullOrEmpty() ||
e.Responsibilities.Trim() == contract.Responsibilities.Trim());
        }

        public IEnumerable<InsuranceAgent> Filter(IEnumerable<InsuranceAgent>
insuranceAgents, InsuranceAgent insuranceAgent) {
            return insuranceAgents.Where(e => insuranceAgent.Salary == 0|| e.Salary
> insuranceAgent.Salary);
        }
        public IEnumerable<InsuranceCase> Filter(IEnumerable<InsuranceCase>
insuranceCases, InsuranceCase insuranceCase) {
            return insuranceCases.Where(e => insuranceCase.InsurancePayment == 0 ||
e.InsurancePayment > insuranceCase.InsurancePayment);
        }

        public IEnumerable<InsuranceType> Filter(IEnumerable<InsuranceType>
insuranceTypes, InsuranceType insuranceType) {
            return insuranceTypes.Where(e => insuranceType.Name.IsNullOrEmpty() ||
e.Name.Trim() == insuranceType.Name.Trim());
        }

        public IEnumerable<Policy> Filter(IEnumerable<Policy> policies, Policy
policy) {
            return policies.Where(e => policy.PolicyPayment == 0 || e.PolicyPayment
> policy.PolicyPayment);
        }

        public IEnumerable<PolicyInsuranceCase>
Filter(IEnumerable<PolicyInsuranceCase> policyInsuranceCases, PolicyInsuranceCase
policyInsuranceCase) {
            return policyInsuranceCases.Where(e => policyInsuranceCase.PolicyId == 0
|| e.PolicyId > policyInsuranceCase.PolicyId);
        }
```

```
        public IEnumerable<SupportingDocument>
Filter(IEnumerable<SupportingDocument> supportingDocuments, SupportingDocument
supportingDocument) {
            return supportingDocuments.Where(e =>
supportingDocument.Name.IsNullOrEmpty() || e.Name.Trim() ==
supportingDocument.Name.Trim());
        }
    }
}
```

Листинг класса *IFilterVisitor*

```
using lab3.Models;

namespace lab3.Services
{
    public interface IFilterVisitor
    {
        IEnumerable<AgentType> Filter(IEnumerable<AgentType> agentTypes, AgentType
agentType);
        IEnumerable<Client> Filter(IEnumerable<Client> clients, Client client);
        IEnumerable<Contract> Filter(IEnumerable<Contract> contracts, Contract
contract);
        IEnumerable<InsuranceAgent> Filter(IEnumerable<InsuranceAgent>
insuranceAgents, InsuranceAgent insuranceAgent);
        IEnumerable<InsuranceCase> Filter(IEnumerable<InsuranceCase> insuranceCases,
InsuranceCase insuranceCase);
        IEnumerable<InsuranceType> Filter(IEnumerable<InsuranceType> insuranceTypes,
InsuranceType insuranceType);
        IEnumerable<Policy> Filter(IEnumerable<Policy> policies, Policy policy);
        IEnumerable<PolicyInsuranceCase> Filter(IEnumerable<PolicyInsuranceCase>
policyInsuranceCases, PolicyInsuranceCase policyInsuranceCase);
        IEnumerable<SupportingDocument> Filter(IEnumerable<SupportingDocument>
supportingDocuments, SupportingDocument supportingDocument);
    }
}
```

Листинг класса *InsuranceCompanyHandlers*

```
using lab3.Data;
using lab3.HtmlParsers;
using lab3.LocalStorage.Cookies;
using lab3.LocalStorage.Sessions;
using lab3.Services;

namespace lab3.Handlers
{
    public class InsuranceCompanyHandlers
    {
        public async void GetInfoPage(HttpContext context)
        {
            string responseContent = new HtmlBuilder()
                .SetTitile("Информация о запросе")
                .AddRequestInfo(context.Request)
                .AddBackMenuButton()
                .Build();

            await context.Response.WriteAsync(responseContent);
        }

        public async void GetMainPage(HttpContext context)
        {
            string header = "Главное меню";
            var tables = new List<string>() {
                "Таблица типов агентов", "Таблица клиентов", "Таблица контрактов",
```

```
                "Таблица страховых агентов", "Таблица страховых случаев", "Таблица
типов срахования",
                "Таблица полисов", "Таблица страховых случаев с полисами", "Таблица
дополнительных документов"
            };
            var search_form1 = new List<string>() {
                "search form1 типов агентов", "search form1 клиентов", "search form1
контрактов",
                "search form1 страховых агентов", "search form1 страховых случаев",
"search form1 типов срахования",
                "search form1 полисов", "search form1 страховых случаев с полисами",
"search form1 дополнительных документов"
            };
            var search_form2 = new List<string>() {
                "search form2 типов агентов", "search form2 клиентов", "search form2
контрактов",
                "search form2 страховых агентов", "search form2 страховых случаев",
"search form2 типов срахования",
                "search form2 полисов", "search form2 страховых случаев с полисами",
"search form2 дополнительных документов"
            };
            var tableNames = InsuranceCompanyContext.DbSetNames;

            List<(string Value, string Url)> list = new() {
                new ("Информация о запросе", "\\info")};

            for (int i = 0; i < tableNames.Count; i++)
            {
                list.Add(new(tables[i], "\\table_" + tableNames[i]));
            }

            for (int i = 0; i < tableNames.Count; i++)
            {
                list.Add(new(search_form1[i], "\\search_form1_" + tableNames[i]));
            }

            for (int i = 0; i < tableNames.Count; i++)
            {
                list.Add(new(search_form2[i], "\\search_form2_" + tableNames[i]));
            }

            string responseContent = new HtmlBuilder()
                .SetTitile("Информация о запросе")
                .AddListWithUrl(header, list)
                .Build();

            await context.Response.WriteAsync(responseContent);
        }

        public async void GetTablePage(HttpContext context, string tableName)
        {
            var cache = context.RequestServices.GetService<InsuranceCompanyCache>();
            var entites = cache.GetEnites(tableName);
            var responseContent = new HtmlBuilder()
                .AddTable(entites)
                .AddBackMenuButton()
                .Build();

            await context.Response.WriteAsync(responseContent);
        }

        public async void GetSearchForm1Page(HttpContext context, string tableName)
        {
            var cache = context.RequestServices.GetService<InsuranceCompanyCache>();
            var sessionsVisitor = new SessionsVisitor();
```

```
            var filter = new CacheFilter();

            var entities = cache.GetEnites(tableName);
            var entity = entities.FirstOrDefault().AcceptLocalData(sessionsVisitor,
context);
            entities = entity.AcceptFilter(filter, entities);

            var responseContent = new HtmlBuilder()
                .AddForm(entity, 1)
                .AddTable(entities)
                .AddBackMenuButton()
                .Build();

            await context.Response.WriteAsync(responseContent);
        }

        public async void GetSearchForm2Page(HttpContext context, string tableName)
        {
            var cache = context.RequestServices.GetService<InsuranceCompanyCache>();
            var cookiesVisitor = new CookiesVisitor();
            var filter = new CacheFilter();

            var entities = cache.GetEnites(tableName);
            var entity = entities.FirstOrDefault().AcceptLocalData(cookiesVisitor,
context);
            entities = entity.AcceptFilter(filter, entities);

            var responseContent = new HtmlBuilder()
                .AddForm(entity, 2)
                .AddTable(entities)
                .AddBackMenuButton()
                .Build();

            await context.Response.WriteAsync(responseContent);
        }
    }
}
```

Листинг класса *HtmlBuilder*

```
using lab3.Models;
using System.Text;


namespace lab3.HtmlParsers
{
    public class HtmlBuilder
    {
        private string _title = "default title";
        private string _body = "";

        public HtmlBuilder SetTitile(string title)
        {
            _title = title;
            return this;
        }

        public HtmlBuilder AddRequestInfo(HttpRequest request)
        {
            var htmlRequestInfo = "<div style=\"text-align: center;\"><H1>Информация
о клиенте</H1>";
            htmlRequestInfo += $"<h3>Сервер: {request.Host}</h3>";
            htmlRequestInfo += $"<h3>Путь: {request.PathBase}</h3>";
            htmlRequestInfo += $"<h3>Протокол: {request.Protocol}</h3>";
            htmlRequestInfo += $"<h3>Метод: {request.Method}</h3>";
```

```
            htmlRequestInfo += $"<h3>Схема: {request.Scheme}</h3>";
            _body += htmlRequestInfo;
            return this;
        }

        public HtmlBuilder AddBackMenuButton()
        {
            _body += "<h3><a href=\"\\\">Главная</a></h3>";
            return this;
        }

        public HtmlBuilder AddListWithUrl(string header, IEnumerable<(string Value,
string Url)> itemsWithUrl)
        {
            var htmlList = $"<div style=\"text-align:
center;\"><H1>{header}</H1><ul>";
            foreach (var item in itemsWithUrl)
            {
                htmlList += $"<li><a href={item.Url}>{item.Value}</a></li>";
            }
            htmlList += "</ul>";

            _body += htmlList;
            return this;
        }

        public HtmlBuilder AddTable(IEnumerable<IEntity> entities)
        {
            var htmlTable = "<div style =\"text-align: center;\"><table
border='1'>";
            var htmlParse = new HtmlTableVisitor();
            foreach (var entity in entities)
            {
                htmlTable += entity.AcceptHtml(htmlParse);
            }
            htmlTable += "</table>";
            _body += htmlTable;
            return this;
        }

        public HtmlBuilder AddForm(IEntity entity, int formType)
        {
            var htmlParser = new HtmlFormVisitor(formType);
            var htmlForm = entity.AcceptHtml(htmlParser);
            _body += htmlForm;
            return this;
        }

        public string Build()
        {
            var htmlPage = new StringBuilder();
            htmlPage.Append($"<HTML><HEAD><TITLE>{_title}</TITLE></HEAD><META http-
equiv='Content-Type' content='text/html; charset=utf-8'/><BODY>");
            htmlPage.Append(_body);
            htmlPage.Append("</div></BODY></HTML>");
            return htmlPage.ToString();
        }
    }
}
```

Листинг класса *HtmlFormVisitor*

```
using lab3.Models;


namespace lab3.HtmlParsers
```

```
{
    public class HtmlFormVisitor : IHtmlVisitor
    {
        private int _formType;

        public HtmlFormVisitor(int formType)
        {
            _formType = formType;
        }

        public string Parse(AgentType agentType)
        {
            return
                $"<form action='/search_form{_formType}_AgentTypes' method='POST'>"
+
                $"Название: <input type='text' name='Type{_formType}' value =
'{agentType.Type}'>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(Client client)
        {
            return
                $"<form action='/search_form{_formType}_Clients' method='POST'>" +
                $"Имя: <input type='text' name='ClientName{_formType}' value =
{client.Name}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(Contract contract)
        {
            return
                $"<form action='/search_form{_formType}_Contracts' method='POST'>" +
                $"Ответственность: <input type='text'
name='Responsibilities{_formType}' value = {contract.Responsibilities}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(InsuranceAgent insuranceAgent)
        {
            return
                $"<form action='/search_form{_formType}_InsuranceAgents'
method='POST'>" +
                $"Зарплата: <input type='number' name='Salary{_formType}' value =
{insuranceAgent.Salary}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(InsuranceCase insuranceCase)
        {
            return
                $"<form action='/search_form{_formType}_InsuranceCases'
method='POST'>" +
                $"Страховая плата: <input type='number'
name='InsurancePayment{_formType}' value = {insuranceCase.InsurancePayment}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(InsuranceType insuranceType)
        {
            return
                $"<form action='/search_form{_formType}_InsuranceTypes'
method='POST'>" +
                $"название: <input type='text' name='InsuranceTypeName{_formType}'
value = {insuranceType.Name}>" +
```

```
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(Policy policy)
        {
            return
                $"<form action='/search_form{_formType}_Policies' method='POST'>" +
                $"стоимость полиса: <input type='text'
name='PolicyPayment{_formType}' value = {policy.PolicyPayment}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(PolicyInsuranceCase policyInsuranceCase)
        {
            return
                $"<form action='/search_form{_formType}_PolicyInsuranceCases'
method='POST'>" +
                $"ID полиса: <input type='number' name='PolicyId{_formType}' value =
{policyInsuranceCase.PolicyId}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }

        public string Parse(SupportingDocument supportingDocument)
        {
            return
                $"<form action='/search_form{_formType}_SupportingDocuments'
method='POST'>" +
                $"название: <input type='text'
name='SupportingDocumentName{_formType}' value = {supportingDocument.Name}>" +
                "<INPUT type ='submit' value='Показать'></FORM>";
        }
    }
}
```

Листинг класса *HtmlTableVisitor*

```
using lab3.Models;

namespace lab3.HtmlParsers
{
    public class HtmlTableVisitor : IHtmlVisitor
    {
        public string Parse(AgentType agentType)
        {
            return
                $"<tr>" +
                    $"<td>{agentType.Id}</td>" +
                    $"<td>{agentType.Type}</td>" +
                $"</tr>";
        }

        public string Parse(Client client)
        {
            return
                $"<tr>" +
                    $"<td>{client.Id}</td>" +
                    $"<td>{client.Name}</td>" +
                    $"<td>{client.Surname}</td>" +
                    $"<td>{client.MiddleName}</td>" +
                    $"<td>{client.Birthdate}</td>" +
                    $"<td>{client.MobilePhone}</td>" +
                    $"<td>{client.City}</td>" +
                    $"<td>{client.Street}</td>" +
                    $"<td>{client.House}</td>" +
                    $"<td>{client.Apartment}</td>" +
                    $"<td>{client.PassportNumber}</td>" +
```

```
            $"<td>{client.PassportIdentification}</td>" +
            $"<td>{client.PassportIssueDate}</td>" +
        $"</tr>";
}

public string Parse(Contract contract)
{
    return
        $"<tr>" +
            $"<td>{contract.Id}</td>" +
            $"<td>{contract.Responsibilities}</td>" +
            $"<td>{contract.StartDeadline}</td>" +
            $"<td>{contract.EndDeadline}</td>" +
        $"</tr>";
}

public string Parse(InsuranceAgent insuranceAgent)
{
    return
        $"<tr>" +
            $"<td>{insuranceAgent.Id}</td>" +
            $"<td>{insuranceAgent.Name}</td>" +
            $"<td>{insuranceAgent.Surname}</td>" +
            $"<td>{insuranceAgent.MiddleName}</td>" +
            $"<td>{insuranceAgent.Salary}</td>" +
            $"<td>{insuranceAgent.TransactionPercent}</td>" +
            $"<td>{insuranceAgent.Contract}</td>" +
            $"<td>{insuranceAgent.AgentType}</td>" +
        $"</tr>";
}

public string Parse(InsuranceCase insuranceCase)
{
    return
        $"<tr>" +
            $"<td>{insuranceCase.Id}</td>" +
            $"<td>{insuranceCase.Client}</td>" +
            $"<td>{insuranceCase.InsuranceAgent}</td>" +
            $"<td>{insuranceCase.Date}</td>" +
            $"<td>{insuranceCase.InsurancePayment}</td>" +
            $"<td>{insuranceCase.SupportingDocument}</td>" +
        $"</tr>";
}

public string Parse(InsuranceType insuranceType)
{
    return
        $"<tr>" +
            $"<td>{insuranceType.Id}</td>" +
            $"<td>{insuranceType.Name}</td>" +
            $"<td>{insuranceType.Description}</td>" +
        $"</tr>";
}

public string Parse(Policy policy)
{
    return
        $"<tr>" +
            $"<td>{policy.Id}</td>" +
            $"<td>{policy.PolicyNumber}</td>" +
            $"<td>{policy.PolicyPayment}</td>" +
            $"<td>{policy.PolicyTerm}</td>" +
            $"<td>{policy.ApplicationDate}</td>" +
            $"<td>{policy.Client}</td>" +
            $"<td>{policy.InsuranceAgent}</td>" +
```

```
                $"<td>{policy.InsuranceType}</td>" +
            $"</tr>";
        }

        public string Parse(PolicyInsuranceCase policyInsuranceCase)
        {
            return
                $"<tr>" +
                    $"<td>{policyInsuranceCase.PolicyId}</td>" +
                    $"<td>{policyInsuranceCase.InsuranceCaseId}</td>" +
                $"</tr>";
        }

        public string Parse(SupportingDocument supportingDocument)
        {
            return
                $"<tr>" +
                    $"<td>{supportingDocument.Id}</td>" +
                    $"<td>{supportingDocument.Name}</td>" +
                    $"<td>{supportingDocument.Description}</td>" +
                $"</tr>";
        }
    }
}
```

Листинг класса *IHtmlVisitor*

```
using lab3.Models;

namespace lab3.HtmlParsers
{
    public interface IHtmlVisitor
    {
        string Parse(AgentType agentType);
        string Parse(Client client);
        string Parse(Contract contract);
        string Parse(InsuranceAgent insuranceAgent);
        string Parse(InsuranceCase insuranceCase);
        string Parse(InsuranceType insuranceType);
        string Parse(Policy policy);
        string Parse(PolicyInsuranceCase policyInsuranceCase);
        string Parse(SupportingDocument supportingDocument);
    }
}
```

Листинг класса *CookiesVisitor*

```
using lab3.Models;

namespace lab3.LocalStorage.Cookies
{
    public class CookiesVisitor : ILocalSaveVisitor
    {
        public AgentType Save(AgentType agentType, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var type = context.Request.Form["Type2"];
                context.Response.Cookies.Append("AgentType", type);
                return new AgentType() { Type = type };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("AgentType"))
                {
```

```
                    return new AgentType() { Type =
context.Request.Cookies["AgentType"] };
                }
                return new AgentType();
            }
        }

        public Client Save(Client client, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var name = context.Request.Form["ClientName2"];
                context.Response.Cookies.Append("Client", name);
                return new Client() { Name = name };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("Client"))
                {
                    return new Client() { Name = context.Request.Cookies["Client"]
};
                }
                return new Client();
            }
        }

        public Contract Save(Contract contract, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var responsibilities = context.Request.Form["Responsibilities2"];
                context.Response.Cookies.Append("Contract", responsibilities);
                return new Contract() { Responsibilities = responsibilities };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("Contract"))
                {
                    return new Contract() { Responsibilities =
context.Request.Cookies["Contract"] };
                }
                return new Contract();
            }
        }

        public InsuranceAgent Save(InsuranceAgent insuranceAgent, HttpContext
context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["Salary2"], out var salary);
                context.Response.Cookies.Append("InsuranceAgent",
salary.ToString());
                return new InsuranceAgent() { Salary = salary };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("InsuranceAgent"))
                {
                    return new InsuranceAgent() { Salary =
decimal.Parse(context.Request.Cookies["InsuranceAgent"]) };
                }
                return new InsuranceAgent();
            }
        }
```

```
        public InsuranceCase Save(InsuranceCase insuranceCase, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["InsurancePayment2"], out var
insurancePayment);
                context.Response.Cookies.Append("InsuranceCase",
insurancePayment.ToString());
                return new InsuranceCase() { InsurancePayment = insurancePayment };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("InsuranceCase"))
                {
                    return new InsuranceCase() { InsurancePayment =
decimal.Parse(context.Request.Cookies["InsuranceCase"]) };
                }
                return new InsuranceCase();
            }
        }

        public InsuranceType Save(InsuranceType insuranceType, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var insuranceTypeName = context.Request.Form["InsuranceTypeName2"];
                context.Response.Cookies.Append("InsuranceType", insuranceTypeName);
                return new InsuranceType() { Name = insuranceTypeName };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("InsuranceType"))
                {
                    return new InsuranceType() { Name =
context.Request.Cookies["InsuranceType"] };
                }
                return new InsuranceType();
            }
        }

        public Policy Save(Policy policy, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["PolicyPayment2"], out var
policyPayment);
                context.Response.Cookies.Append("Policy", policyPayment.ToString());
                return new Policy() { PolicyPayment = policyPayment };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("Policy"))
                {
                    return new Policy() { PolicyPayment =
decimal.Parse(context.Request.Cookies["Policy"]) };
                }
                return new Policy();
            }
        }

        public PolicyInsuranceCase Save(PolicyInsuranceCase policyInsuranceCase,
HttpContext context)
        {
            if (context.Request.Method == "POST")
```

```
            {
                int.TryParse(context.Request.Form["PolicyId2"], out var policyId);
                context.Response.Cookies.Append("PolicyInsuranceCase",
policyId.ToString());
                return new PolicyInsuranceCase() { PolicyId = policyId };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("PolicyInsuranceCase"))
                {
                    return new PolicyInsuranceCase() { PolicyId =
int.Parse(context.Request.Cookies["PolicyInsuranceCase"]) };
                }
                return new PolicyInsuranceCase();
            }
        }

        public SupportingDocument Save(SupportingDocument supportingDocument,
HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var name = context.Request.Form["SupportingDocumentName2"];
                context.Response.Cookies.Append("SupportingDocument", name);
                return new SupportingDocument() { Name = name };
            }
            else
            {
                if (context.Request.Cookies.ContainsKey("SupportingDocument"))
                {
                    return new SupportingDocument() { Name =
context.Request.Cookies["SupportingDocument"] };
                }
                return new SupportingDocument();
            }
        }
    }
}
```

Листинг класса *SessionExtensions*

```
using Newtonsoft.Json;

namespace lab3.LocalStorage.Sessions
{
    public static class SessionExtensions
    {
        public static void Set<T>(this ISession session, string key, T value)
        {
            session.SetString(key, JsonConvert.SerializeObject(value));
        }

        public static T Get<T>(this ISession session, string key)
        {
            var value = session.GetString(key);
            var entity = value == null ? default :
JsonConvert.DeserializeObject<T>(value);
            return entity;
        }
    }
}
```

Листинг класса *SessionsVisitor*

```
using lab3.Models;
```

```
namespace lab3.LocalStorage.Sessions
{
    public class SessionsVisitor : ILocalSaveVisitor
    {
        public AgentType Save(AgentType agentType, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var type = context.Request.Form["Type1"];
                var newAgentType = new AgentType { Type = type };
                context.Session.Set("AgentType", newAgentType);
                return newAgentType;
            }
            else
            {
                agentType = context.Session.Get<AgentType>("AgentType") ?? new
AgentType();
                return agentType;
            }
        }

        public Client Save(Client client, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var name = context.Request.Form["ClientName1"];
                var newClient = new Client { Name = name };
                context.Session.Set("Client", newClient);
                return newClient;
            }
            else
            {
                client = context.Session.Get<Client>("Client") ?? new Client();
                return client;
            }
        }

        public Contract Save(Contract contract, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var responsibilities = context.Request.Form["Responsibilities1"];
                var newClient = new Contract { Responsibilities = responsibilities
};
                context.Session.Set("Contract", newClient);
                return newClient;
            }
            else
            {
                contract = context.Session.Get<Contract>("Contract") ?? new
Contract();
                return contract;
            }
        }

        public InsuranceAgent Save(InsuranceAgent insuranceAgent, HttpContext
context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["Salary1"], out var salary);
                var newInsuranceAgent = new InsuranceAgent { Salary = salary };
                context.Session.Set("InsuranceAgent", newInsuranceAgent);
                return newInsuranceAgent;
            }
```

```
            else
            {
                insuranceAgent =
context.Session.Get<InsuranceAgent>("InsuranceAgent") ?? new InsuranceAgent();
                return insuranceAgent;
            }
        }

        public InsuranceCase Save(InsuranceCase insuranceCase, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["InsurancePayment1"], out var
insurancePayment);
                var newInsuranceCase = new InsuranceCase { InsurancePayment =
insurancePayment };
                context.Session.Set("InsuranceCase", newInsuranceCase);
                return newInsuranceCase;
            }
            else
            {
                insuranceCase = context.Session.Get<InsuranceCase>("InsuranceCase")
?? new InsuranceCase();
                return insuranceCase;
            }
        }

        public InsuranceType Save(InsuranceType insuranceType, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var name = context.Request.Form["InsuranceTypeName1"];
                var newInsuranceType = new InsuranceType { Name = name };
                context.Session.Set("InsuranceType", newInsuranceType);
                return newInsuranceType;
            }
            else
            {
                insuranceType = context.Session.Get<InsuranceType>("InsuranceType")
?? new InsuranceType();
                return insuranceType;
            }
        }

        public Policy Save(Policy policy, HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                decimal.TryParse(context.Request.Form["PolicyPayment1"], out var
policyPayment);
                var newPolicy = new Policy { PolicyPayment = policyPayment };
                context.Session.Set("Policy", newPolicy);
                return newPolicy;
            }
            else
            {
                policy = context.Session.Get<Policy>("Policy") ?? new Policy();
                return policy;
            }
        }

        public PolicyInsuranceCase Save(PolicyInsuranceCase policyInsuranceCase,
HttpContext context)
        {
            if (context.Request.Method == "POST")
```

```
            {
                int.TryParse(context.Request.Form["PolicyId1"], out var policyId);
                var newPolicyInsuranceCase = new PolicyInsuranceCase { PolicyId =
policyId };
                context.Session.Set("PolicyInsuranceCase", newPolicyInsuranceCase);
                return newPolicyInsuranceCase;
            }
            else
            {
                policyInsuranceCase =
context.Session.Get<PolicyInsuranceCase>("PolicyInsuranceCase") ?? new
PolicyInsuranceCase();
                return policyInsuranceCase;
            }
        }

        public SupportingDocument Save(SupportingDocument supportingDocument,
HttpContext context)
        {
            if (context.Request.Method == "POST")
            {
                var name = context.Request.Form["SupportingDocumentName1"];
                var newSupportingDocument = new SupportingDocument { Name = name };
                context.Session.Set("SupportingDocument", newSupportingDocument);
                return newSupportingDocument;
            }
            else
            {
                supportingDocument =
context.Session.Get<SupportingDocument>("SupportingDocument") ?? new
SupportingDocument();
                return supportingDocument;
            }
        }
    }
}
```

Листинг класса *ILocalSaveVisitor*

```
using lab3.Models;

namespace lab3.LocalStorage
{
    public interface ILocalSaveVisitor
    {
        AgentType Save(AgentType agentType, HttpContext context);
        Client Save(Client client, HttpContext context);
        Contract Save(Contract contract, HttpContext context);
        InsuranceAgent Save(InsuranceAgent insuranceAgent, HttpContext context);
        InsuranceCase Save(InsuranceCase insuranceCase, HttpContext context);
        InsuranceType Save(InsuranceType insuranceType, HttpContext context);
        Policy Save(Policy policy, HttpContext context);
        PolicyInsuranceCase Save(PolicyInsuranceCase policyInsuranceCase,
HttpContext context);
        SupportingDocument Save(SupportingDocument supportingDocument, HttpContext
context);
    }
}
```

Листинг класса *AgentType*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;
```

```
public partial class AgentType : IEntity {
    public int Id { get; set; }

    public string Type { get; set; } = null!;

    public virtual ICollection<InsuranceAgent> InsuranceAgents { get; set; } = new
List<InsuranceAgent>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (AgentType)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *Client*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class Client : IEntity {
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string Surname { get; set; } = null!;

    public string MiddleName { get; set; } = null!;

    public DateTime Birthdate { get; set; }

    public string MobilePhone { get; set; } = null!;

    public string City { get; set; } = null!;

    public string Street { get; set; } = null!;

    public string House { get; set; } = null!;

    public string Apartment { get; set; } = null!;

    public string PassportNumber { get; set; } = null!;

    public DateTime PassportIssueDate { get; set; }

    public string PassportIdentification { get; set; } = null!;

    public virtual ICollection<InsuranceCase> InsuranceCases { get; set; } = new
List<InsuranceCase>();

    public virtual ICollection<Policy> Policies { get; set; } = new List<Policy>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (Client)e), this);
    }
```

```
    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *Contract*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class Contract : IEntity {
    public int Id { get; set; }

    public string Responsibilities { get; set; } = null!;

    public DateTime StartDeadline { get; set; }

    public DateTime EndDeadline { get; set; }

    public virtual ICollection<InsuranceAgent> InsuranceAgents { get; set; } = new
List<InsuranceAgent>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (Contract)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *IEntity*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models
{
    public interface IEntity {
        public string AcceptHtml(IHtmlVisitor visitor);
        public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext
context);
        public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities);
    }
}
```

Листинг класса *InsuranceAgent*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;
```

```
namespace lab3.Models;

public partial class InsuranceAgent : IEntity {
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string Surname { get; set; } = null!;

    public string MiddleName { get; set; } = null!;

    public int AgentType { get; set; }

    public decimal Salary { get; set; }

    public int Contract { get; set; }

    public double TransactionPercent { get; set; }

    public virtual AgentType AgentTypeNavigation { get; set; } = null!;

    public virtual Contract ContractNavigation { get; set; } = null!;

    public virtual ICollection<InsuranceCase> InsuranceCases { get; set; } = new
List<InsuranceCase>();

    public virtual ICollection<Policy> Policies { get; set; } = new List<Policy>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (InsuranceAgent)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *InsuranceCase*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class InsuranceCase : IEntity {
    public int Id { get; set; }

    public int Client { get; set; }

    public int InsuranceAgent { get; set; }

    public DateTime Date { get; set; }

    public string? Description { get; set; }

    public int SupportingDocument { get; set; }

    public decimal InsurancePayment { get; set; }
```

```
    public virtual Client ClientNavigation { get; set; } = null!;

    public virtual InsuranceAgent InsuranceAgentNavigation { get; set; } = null!;

    public virtual SupportingDocument SupportingDocumentNavigation { get; set; } =
null!;

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (InsuranceCase)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *InsuranceType*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class InsuranceType : IEntity {
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string? Description { get; set; }

    public virtual ICollection<Policy> Policies { get; set; } = new List<Policy>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (InsuranceType)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *Policy*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class Policy : IEntity {
    public int Id { get; set; }

    public int InsuranceAgent { get; set; }

    public DateTime ApplicationDate { get; set; }
```

```
    public string PolicyNumber { get; set; } = null!;

    public int InsuranceType { get; set; }

    public int Client { get; set; }

    public int PolicyTerm { get; set; }

    public decimal PolicyPayment { get; set; }

    public virtual Client ClientNavigation { get; set; } = null!;

    public virtual InsuranceAgent InsuranceAgentNavigation { get; set; } = null!;

    public virtual InsuranceType InsuranceTypeNavigation { get; set; } = null!;

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (Policy)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *PolicyInsuranceCase*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
using lab3.Services;

namespace lab3.Models;

public partial class PolicyInsuranceCase : IEntity {
    public int PolicyId { get; set; }

    public int InsuranceCaseId { get; set; }

    public virtual InsuranceCase InsuranceCase { get; set; } = null!;

    public virtual Policy Policy { get; set; } = null!;

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (PolicyInsuranceCase)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *SupportingDocument*

```
using lab3.HtmlParsers;
using lab3.LocalStorage;
```

```
using lab3.Services;

namespace lab3.Models;

public partial class SupportingDocument : IEntity
{
    public int Id { get; set; }

    public string Name { get; set; } = null!;

    public string? Description { get; set; }

    public virtual ICollection<InsuranceCase> InsuranceCases { get; set; } = new
List<InsuranceCase>();

    public IEnumerable<IEntity> AcceptFilter(IFilterVisitor visitor,
IEnumerable<IEntity> entities) {
        return visitor.Filter(entities.Select(e => (SupportingDocument)e), this);
    }

    public string AcceptHtml(IHtmlVisitor visitor) {
        return visitor.Parse(this);
    }

    public IEntity AcceptLocalData(ILocalSaveVisitor visitor, HttpContext context) {
        return visitor.Save(this, context);
    }
}
```

Листинг класса *Program*

```
using lab3.Data;
using lab3.Handlers;
using Microsoft.EntityFrameworkCore;


internal class Program {
    private static void Main(string[] args) {
        var builder = WebApplication.CreateBuilder(args);

        builder.Services.AddDbContext<InsuranceCompanyContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("InsuranceCompany")))
;
        builder.Services.AddTransient<InsuranceCompanyCache>();
        builder.Services.AddMemoryCache();
        builder.Services.AddDistributedMemoryCache();
        builder.Services.AddSession();

        var app = builder.Build();
        app.UseSession();
        app.UseCookiePolicy();

        var handler = new InsuranceCompanyHandlers();
        var tables = InsuranceCompanyContext.DbSetNames;


        app.Map("/info", (appBuilder) => appBuilder.Run(async (context) =>
handler.GetInfoPage(context)));
        foreach (var table in tables){
            app.Map($"/table_{table}", (appBuilder) => appBuilder.Run(async
(context) => handler.GetTablePage(context, table)));
        }
        foreach (var table in tables) {
            app.Map($"/search_form1_{table}", (appBuilder) => appBuilder.Run(async
(context) => handler.GetSearchForm1Page(context, table)));
        }
```

```
        foreach (var table in tables) {
            app.Map($"/search_form2_{table}", (appBuilder) => appBuilder.Run(async
(context) => handler.GetSearchForm2Page(context, table)));
        }

        app.Run(async (context) => handler.GetMainPage(context));

        app.Run();
    }
}
```