

SMP BANK - Russia, Moscow (<http://smpbank.ru/en>).

Project of ABS (internal automated banking system): development UI based on Angular 2 (typescript, PrimeNG components, webpack 2), web services for UI based on asp.net web api (c#).

## Content:

1. UI screenshots
2. Project structure

### 1. UI screenshots

List form example (template view):

Ковальчук Зинаида Петровна

Портфель клиента - кредиты

Кредиты Вклады

☐ Показать закрытые

Номер кредитного договора	Тип продукта	Сумма выдачи	Остаток долга
КД-800-0/0000/2017-0277	Потребительский кредит	100 000,00 RUR	100 000,00 RUR
Статус: Новый	Состояние: Не выдан	Ставка: 12 %	Ближайший платеж: 10.06.2017
+ Подробности			

Номер кредитного договора	Тип продукта	Сумма выдачи	Остаток долга
КД-800-0/0000/2017-0279	Потребительский кредит	100 000,00 RUR	100 000,00 RUR
Статус: Новый	Состояние: Не выдан	Ставка: 12 %	Ближайший платеж: 10.06.2017
+ Подробности			

Номер кредитного договора	Тип продукта	Сумма выдачи	Остаток долга
КД-800-0/0000/2017-0280	Потребительский кредит	150 000,00 RUR	150 000,00 RUR
Статус: Новый	Состояние: Не выдан	Ставка: 12 %	Ближайший платеж: 10.06.2017
+ Подробности			

List form example (table view):

Поиск клиента

Расширенный поиск...

Операции

Обработка

Мониторинг

Поручения

Итоги дня

Регулярные задачи

Настройки

Статистика

Список поручений

Фильтр

Статус поручения: Не выбрано

Операция: Не выбрано

Номер клиента

Подразделение

Дата начала

Дата окончания

Показать

показано 100 записей на странице

#	Дата	Клиент	Операция	Статус	Сумма
8406	15.06.2017 17:28:40	1 - Ковальчук Зинаида Петровна	0000 - Тестовая операция	Активно	
8403	15.06.2017 17:28:39	1 - Ковальчук Зинаида Петровна	0000 - Тестовая операция	Аннулируется	
8402	15.06.2017 17:28:36	1 - Ковальчук Зинаида Петровна	0000 - Тестовая операция	Активно	
8399	15.06.2017 17:21:29	1 - Ковальчук Зинаида Петровна	0000 - Тестовая операция	Активно	
8397	15.06.2017 17:21:27	1 - Ковальчук Зинаида Петровна	0000 - Тестовая операция	Аннулируется	

Edit form example:

**ФИНИК** Создание счета

Поиск клиента    
Расширенный поиск...

Операции  
Обработка  
Мониторинг  
Настройки  
Счета  
Счета  
Режимы  
Периоды  
Статистика

Справочник счетов

Наименование счёта

Номер счёта **Не выбрано**   
Обязательное поле

Режим **Не выбрано**   
Обязательное поле

Резидентство **Не выбрано**   
Обязательное поле

Тип счёта **Не выбрано**   
Обязательное поле

Классификатор **Не выбрано**   
Обязательное поле

Валюта **Не выбрано**   
Обязательное поле

Период **Не выбрано**   
Обязательное поле

Дата открытия счёта

☐ Ведение и контроль остатков ☐ Создавать счет второго баланса

Подразделение

Ассоциации

Ассоциированная валюта **Не выбрано**   
Ассоциированное подразделение

[← В справочник счетов](#) Сохранить

Группа СМП Банк © 2017 Сделано с

Business-operations menu form example:

**ФИНИК** Список операций

Поиск клиента    
Расширенный поиск...

Операции  
Обработка  
Мониторинг  
Настройки  
Статистика

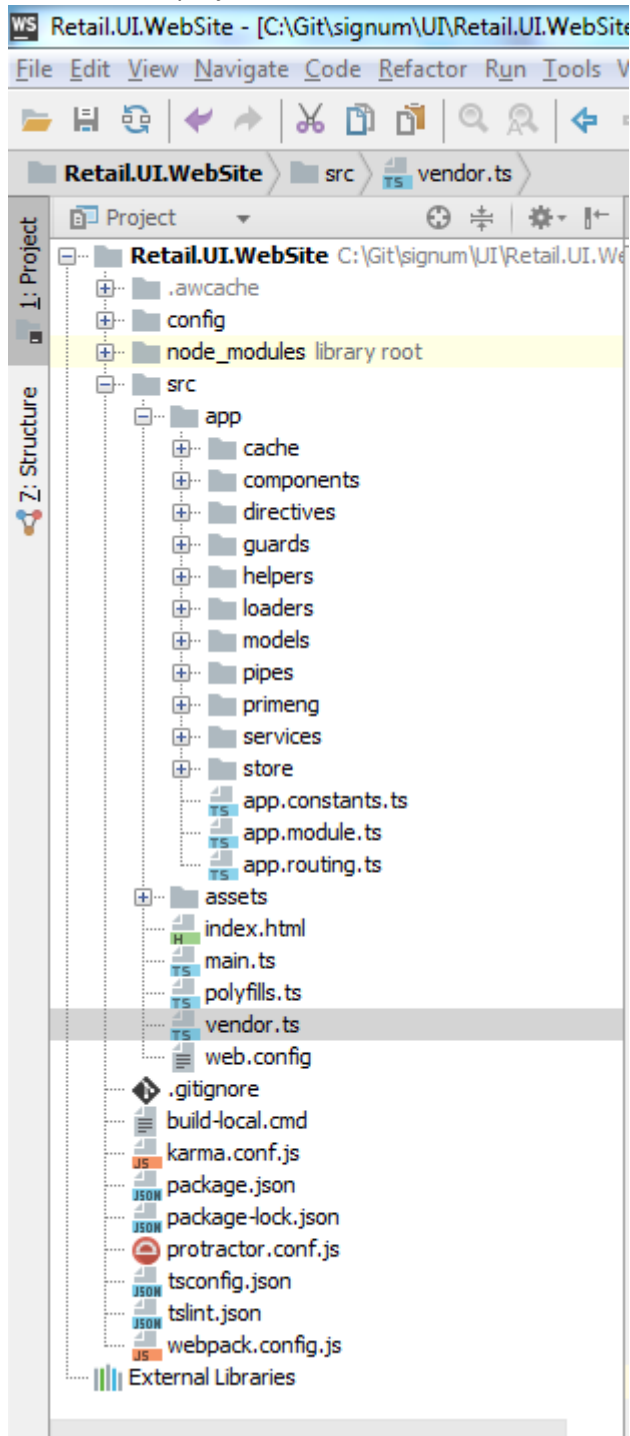
Операции по кредитам

- ▶ 0000 - Тестовая операция
- ▶ 0001 - Открытие счета
- ▶ 2010 - Перевод между счетами
- ▶ 2302 - Обслуживание долга по кредитам
- ▶ 2303 - Обслуживание индивидуальных резервов кредита
- ▶ 2304 - Корректировка резервов ПОС
- ▶ 2305 - Изменение категории качества индивидуальных резервов кредита
- ▶ 2306 - Реклассификация кредита
- ▶ 2307 - Изменение типа резерва кредита

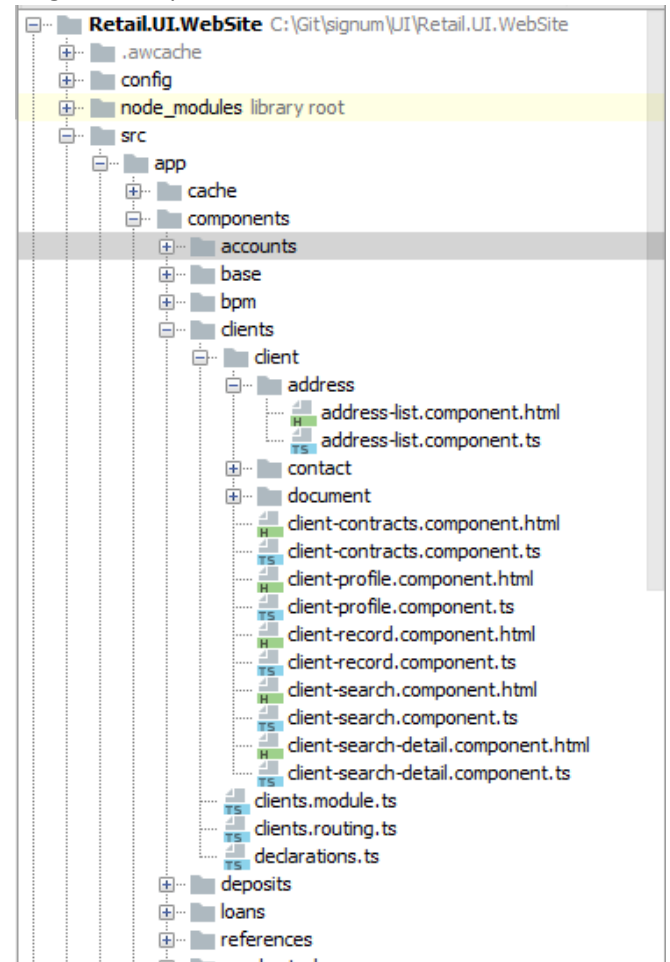
Группа СМП Банк © 2017 Сделано с

## 2. Project structure

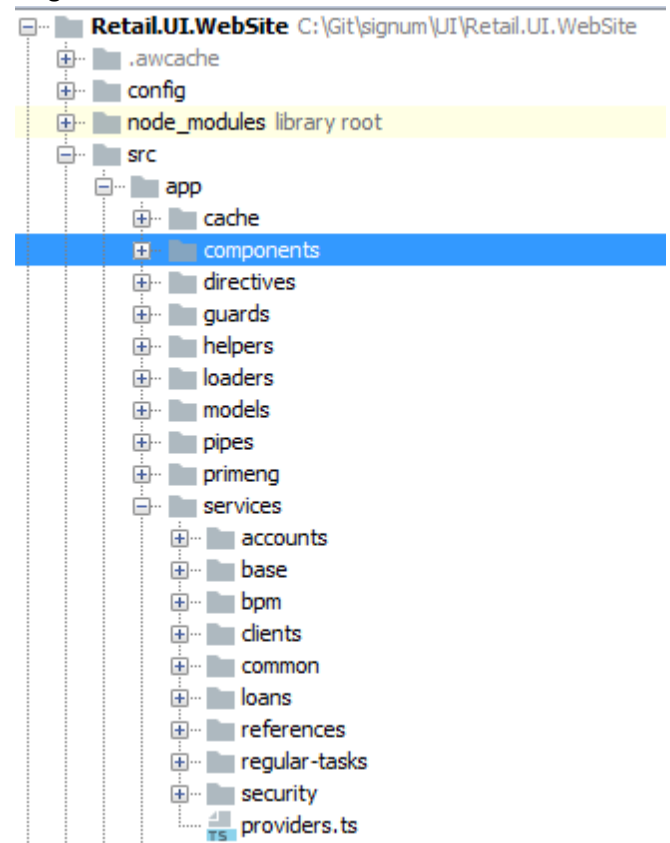
WebStorm UI project structure:



Angular Components structure:



Angular Services structure:



## Angular Service code example:

```
import { Injectable } from '@angular/core';
import { ApiService } from '../common/api.service';

import { AccountGetDemand } from '../../models/accounts/account-get-demand';
import { Account } from '../../models/accounts/account';
import { AccountCreateDemand } from '../../models/accounts/account-create-demand';
import { AccountCloseDemand } from '../../models/accounts/account-close-demand';
import { AccountUpdateDemand } from '../../models/accounts/account-update-demand';

const apiRoutePrefix = 'api/accounts/account';

/** Api for technical accounts */
@Injectable()
export class AccountService {

  constructor(private apiService: ApiService) {
  }

  /** return entity list by filter */
  public list(filter: AccountGetDemand): Promise<Account[]> {
    return this.apiService.post({
      urlParts: [apiRoutePrefix, 'list'],
      data: filter,
      metadata: {type: AccountGetDemand},
      result: {type: Array, elements: {type: Account}},
    });
  }

  /** return entity by id */
  public detail(id: number): Promise<Account> {
    return this.apiService.get({
      urlParts: [apiRoutePrefix, 'detail', id],
      result: {type: Account},
    });
  }

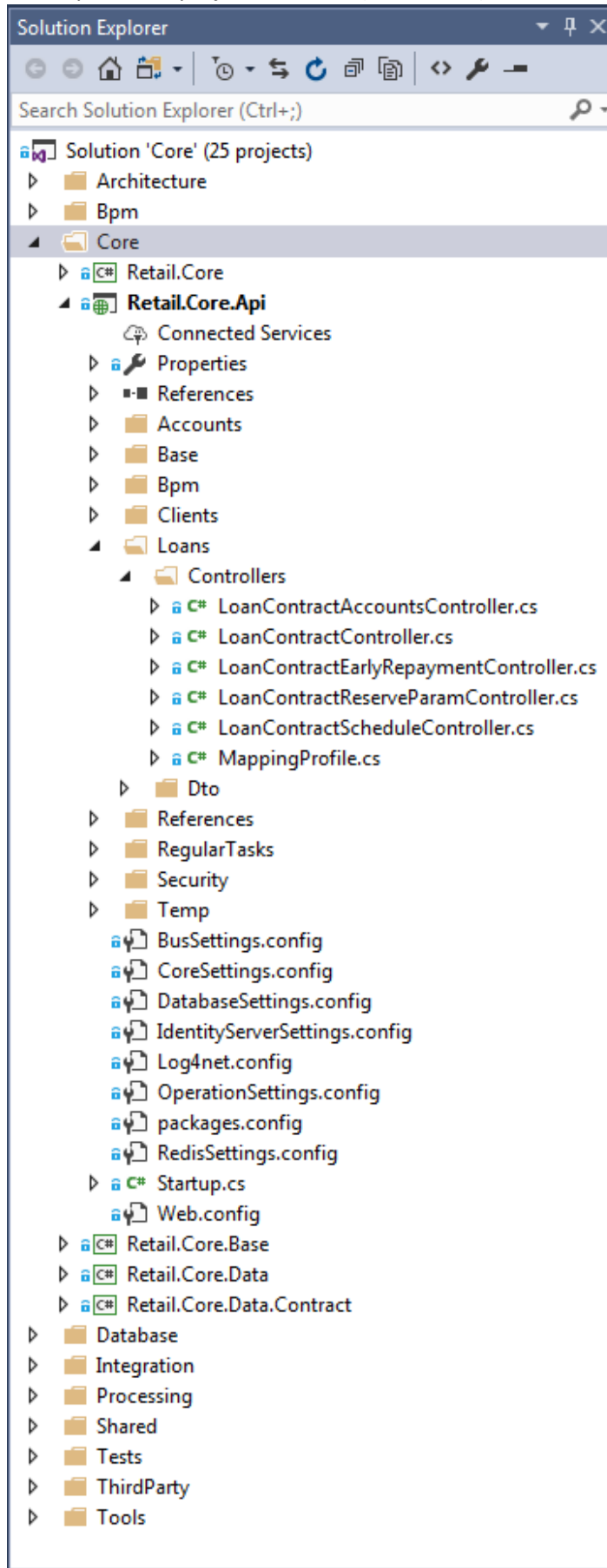
  /** create entity */
  public create(entity: AccountCreateDemand): Promise<Account> {
    return this.apiService.post({
      urlParts: [apiRoutePrefix, 'create'],
      data: entity,
      metadata: {type: AccountCreateDemand},
      result: {type: Account},
    });
  }

  /** delete entity */
  public remove(id: number): Promise<void> {
    return this.apiService.post({
      urlParts: [apiRoutePrefix, 'remove', id],
    });
  }

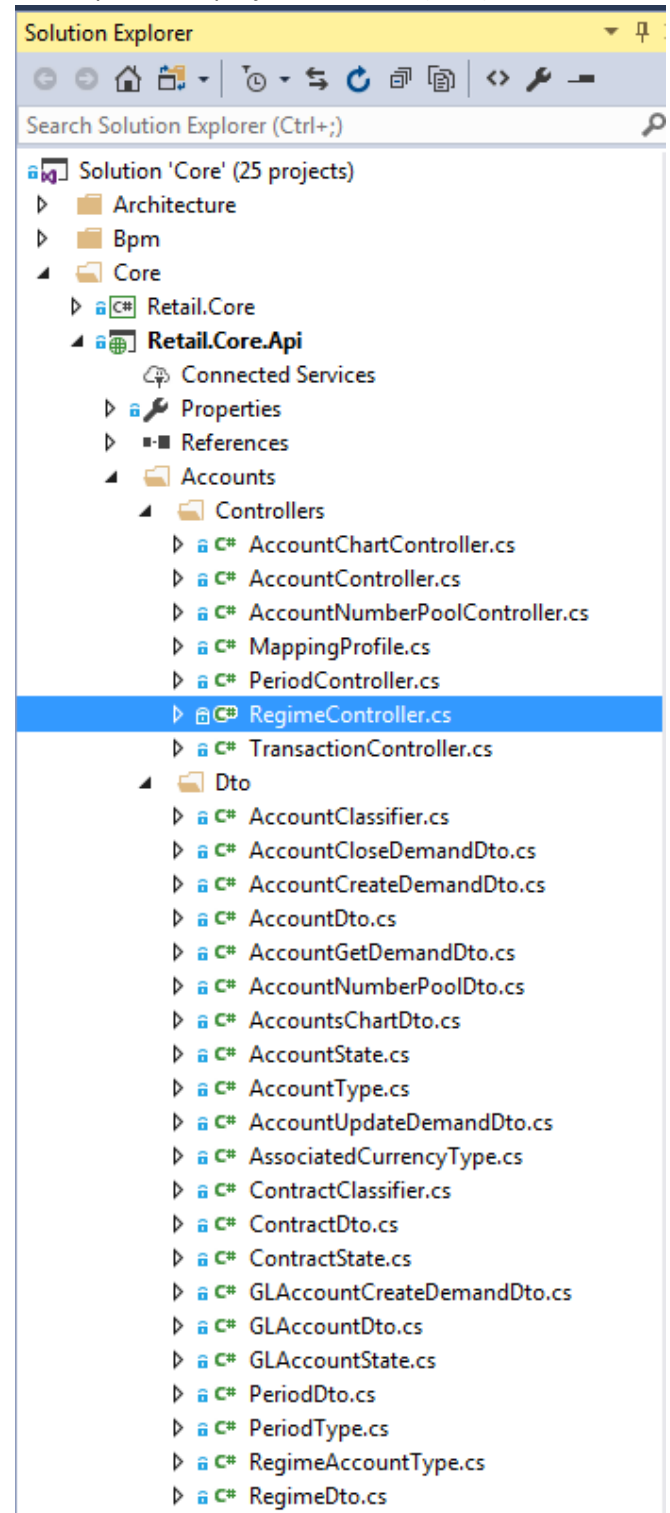
  /** specialized operation - close entity */
  public close(entity: AccountCloseDemand): Promise<Account> {
    return this.apiService.post({
      urlParts: [apiRoutePrefix, 'close'],
      data: entity,
      metadata: {type: AccountCloseDemand},
      result: {type: Account},
    });
  }

  /** update entity */
  public update(entity: AccountUpdateDemand): Promise<Account> {
    return this.apiService.post({
      urlParts: [apiRoutePrefix, 'update'],
      data: entity,
      metadata: {type: AccountUpdateDemand},
      result: {type: Account},
    });
  }
}
```

### Web-api service project structure (controllers):



### Web-api service project structure (DTO):



## Web-api service code example:

```
using System;
using System.Threading.Tasks;
using System.Web.Http;
using AutoMapper;
using log4net;
using Retail.Core.Accounts.Data;
using Retail.Core.Accounts.Services;
using Retail.Core.Api.Accounts.Dto;
using Retail.Core.Api.Base.Controllers;

namespace Retail.Core.Api.Accounts.Controllers
{
    /// <summary> Api режимов счетов </summary>
    [RoutePrefix("api/accounts/regime")]
    public class RegimeController : RetailApiControllerBase
    {
        private readonly IRegimeService _service;
        private readonly ILog _log = LogManager.GetLogger(typeof(Startup));

        public RegimeController(IRegimeService service)
        {
            _service = service;
        }

        /// <summary> Возвращает список сущностей </summary>
        [HttpGet]
        [Route("list")]
        public async Task<RegimeDto[]> List()
        {
            return await Task.Run<RegimeDto[]>(() =>
            {
                try
                {
                    //throw new Exception("Веб-сервис вернул Exception");

                    var result = _service.GetAll();
                    return Mapper.Map<RegimeDto[]>(result);
                }
                catch (Exception e)
                {
                    throw HttpResponseException(e);
                }
            });
        }

        /// <summary> Возвращает заданную сущность </summary>
        [HttpGet]
        [Route("detail/{id}")]
        public async Task<RegimeDto> Detail([FromUri] int id)
        {
            return await Task.Run<RegimeDto>(() =>
```

```

        {
            try
            {
                //throw new Exception("Веб-сервис вернул Exception");

                var result = _service.Get(id);
                return Mapper.Map<RegimeDto>(result);
            }
            catch (Exception e)
            {
                throw HttpResponseException(e);
            }
        });
    }

    /// <summary> Обновляет сущность </summary>
    [HttpPost]
    [Route("update")]
    public async Task<RegimeDto> Update([FromBody] RegimeDto regimeDto)
    {
        return await Task.Run<RegimeDto>(() =>
        {
            try
            {
                //throw new Exception("Веб-сервис вернул Exception");

                /*
                throw new UserException(new List<UserMessage>()
                {
                    new UserMessage(fieldName: "Name", message: "Наименование - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                    new UserMessage(fieldName: "MinLengthDays", message: "MinLengthDays - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                    new UserMessage(fieldName: "MinLengthMonths", message: "MinLengthMonths - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                    new UserMessage(fieldName: "MaxLengthDays", message: "MaxLengthDays - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                    new UserMessage(fieldName: "MaxLengthMonths", message: "MaxLengthMonths - Validation error", level: UserMessage.MessageLevel.Error, code: "0")
                });
                */

                var regimeInput = Mapper.Map<Regime>(regimeDto);
                var regimeOutput = _service.Update(regimeInput);
                var regimeDtoOutput = Mapper.Map<RegimeDto>(regimeOutput);
                return regimeDtoOutput;
            }
            catch (Exception e)
            {
                throw HttpResponseException(e);
            }
        });
    }

    /// <summary> Создаёт сущность </summary>
    [HttpPost]
    [Route("create")]
    //[ResponseType(typeof(RegimeDto))]
    //public async Task<IHttpActionResult> Create([FromBody] RegimeDto regimeDto)

```

```

public async Task<RegimeDto> Create([FromBody] RegimeDto regimeDto)
{
    return await Task.Run<RegimeDto>(() =>
    {
        try
        {
            //throw new Exception("Веб-сервис вернул Exception");

            /*
            throw new UserException(new List<UserMessage>()
            {
                new UserMessage(fieldName: "Name", message: "Наименование - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                new UserMessage(fieldName: "MinLengthDays", message: "MinLengthDays - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                new UserMessage(fieldName: "MinLengthMonths", message: "MinLengthMonths - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                new UserMessage(fieldName: "MaxLengthDays", message: "MaxLengthDays - Validation error", level: UserMessage.MessageLevel.Error, code: "0"),
                new UserMessage(fieldName: "MaxLengthMonths", message: "MaxLengthMonths - Validation error", level: UserMessage.MessageLevel.Error, code: "0")
            }));
            */

            var regimeInput = Mapper.Map<Regime>(regimeDto);
            var regimeOutput = _service.Create(regimeInput);
            var regimeDtoOutput = Mapper.Map<RegimeDto>(regimeOutput);

            return regimeDtoOutput;
        }
        catch (Exception e)
        {
            throw HttpResponseException(e);
        }
    });
}

/// <summary> Удаляет сущность </summary>
[HttpPost]
[Route("remove/{id}")]
public async Task Remove([FromUri] int id)
{
    await Task.Run(() =>
    {
        try
        {
            _service.Delete(id);
        }
        catch (Exception e)
        {
            throw HttpResponseException(e);
        }
    });
}
}
}

```



Web-api DTO code example:

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Retail.Core.Api.References.Dto;

namespace Retail.Core.Api.Accounts.Dto
{
    /// <summary>
    /// Определение ограничений счетов
    /// </summary>
    public class RegimeDto
    {
        #region Simple columns

        /// <summary>
        /// ИД
        /// </summary>
        [Display(Name = "ИД"), Required]
        public int Id { get; set; } // int

        /// <summary>
        /// Наименование
        /// </summary>
        [Display(Name = "Наименование"), MaxLength(500), Required]
        public string Name { get; set; } // varchar(500)

        /// <summary>
        /// Счёт: Уровень организационной структуры
        /// </summary>
        [Display(Name = "Счёт: Уровень организационной структуры"), Required]
        public OrgUnitType OrgUnitType { get; set; } // smallint

        /// <summary>
        /// Счёт: Классификатор (клиентский, системный и т.д.)
        /// </summary>
        [Display(Name = "Счёт: Классификатор (клиентский, системный и т.д.)"), Required]
        public AccountClassifier AccountClassifier { get; set; } // smallint

        /// <summary>
        /// Счёт: Резиденство
        /// </summary>
        [Display(Name = "Счёт: Резиденство"), Required]
        public RegimeResidenceType AccountResidenceType { get; set; } // smallint
    }
}
```

```

/// <summary>
/// Счёт: Тип (Активный/Пассивный)
/// </summary>
[Display(Name = "Счёт: Тип (Активный/Пассивный)", Required)]
public RegimeAccountType AccountType { get; set; } // smallint

/// <summary>
/// Счёт: Уникален ли по характеристикам
/// </summary>
[Display(Name = "Счёт: Уникален ли по характеристикам"), Required]
public bool IsAccountUnique { get; set; } // bit

/// <summary>
/// Счёт: Маска наименования
/// </summary>
[Display(Name = "Счёт: Маска наименования"), MaxLength(100)]
public string AccountNameMask { get; set; } // varchar(100)

/// <summary>
/// Доступен
/// </summary>
[Display(Name = "Доступен"), Required]
public bool IsEnabled { get; set; } // bit

/// <summary>
/// Связанный объект: Устанавливается ли валюта
/// </summary>
[Display(Name = "Связанный объект: Устанавливается ли валюта"), Required]
public bool SetAssociatedCurrency { get; set; } // bit

/// <summary>
/// Связанный объект: Уровень организационной структуры
/// </summary>
[Display(Name = "Связанный объект: Уровень организационной структуры")]
public OrgUnitType? AssociatedOrgUnitType { get; set; } // smallint

#endregion

#region Relations

/// <summary>
/// План счетов для режима
/// </summary>
[Display(Name = "План счетов для режима")]
public IEnumerable<int> RegimesAccountsChart { get; set; }

```

```
/// <summary>
/// Валюты для режима
/// </summary>
[Display(Name = "Валюты для режима")]
public IEnumerable<int> RegimesCurrencies { get; set; }
```

```
#endregion
```

```
}
```

```
}
```