Министерство науки и высшего образования Российской Федерации
ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»
Кафедра «школа бакалавриата (школа)»

Тема задания на практику
Изучение технологий создания IT приложений

ОТЧЕТ

Вид практики Производственная практика
Тип практики Технологическая практика

Руководитель практики от предприятия (организации): **ФИО руководителя** **Подпись**

Студент: Бутузов А.В.
ФИО студента

Специальность (направление подготовки): 09.03.01 Информатика и вычислительная техника

Группа: РИ-370017

Екатеринбург 2020

Содержание

Введение

В работе многим компаниям часто нужно прогнозировать какие-либо данные и ситуации, моделировать процессы и испытывать новые наработки. Например, когда нужно узнать слабые места в производстве, или же когда вводится новый аппарат, и нужно узнать новую скорость работы. Для этого используются специальные программы для моделирования систем, которые позволяют увидеть, как статистические данные работы, так и визуальный опыт. AnyLogic является примером программного обеспечения для имитационного моделирования.

Однако, порой требуются программы, которые предназначены для моделирования какой-то конкретной ситуации для конкретного предприятия. Пример такого программного обеспечения выполнялся на практике. Была дана сеть супермаркетов, которая реализует некоторое число продуктов, число которых было ограничено. Надо было написать программу, которая моделирует работу этих супермаркетов, учитывая средняя время обслуживания на кассах, количество товара и количество людей в разный промежуток времени, и показывает, когда тот или иной товар закончился или близок к концу. Программа показывает, как табличные статистические данные, так и график работы.

Постановка задачи

Имеется Сеть супермаркетов (12 магазинов), в которой реализуются 8 видов основных товаров. Каждый вид товара характеризуется сроком хранения и объемом, перевозимым 1 грузовиком.

A. молочная продукция: 3 дня, 1000 л;

B. хлебобулочные изделия: 3 дня, 800 штук;

C. овощи-фрукты: 3 недели, 1200 кг;

D. мясо, рыба: 3 месяца, 1200 кг;

E. полуфабрикаты: 3 дня, 1200 кг;

F. алкогольная продукция: 12 месяцев, 1000 бутылок;

G. соки, воды: 12 месяцев, 1000 упаковок.

Поставка товаров осуществляется типовым транспортным средством. Каждый вид товаров транспортируется отдельно от других видов. Известны данные о средней потребительской корзине и интенсивность потока покупателей. Склады супермаркетов имеют ограничения. Просроченные товары подлежат утилизации (возвращению поставщику).

Исходные данные:

1) Потребительская корзина: средний набор продуктов клиента по магазинам (помер вида товара - количество):

магазины 1-3: A - 0,5 л. B - 0,3 шт. C - 1 кг, D - 1 кг. E - 1 кг, F - 0,2 бут. G - 1 уп.

магазины 4-7: A - 1 л. B - 1 шт. C - 1,9 кг, D - 0,4 кг, E - 1 кг, F- 0,4 бут. G - 0,3 уп.

магазины 8-12: - 0,7 л. B - 0,7 шт. C - 2,5 кг, D - 1,5 кг, E - 1 кг, F - 0,5 бут. G - 0,8 уп

2) Интенсивность прихода клиентов:

утро (8-13) 100+20 человек в час,

день (13-17) 150-30 человек в час,

вечер (17-21) 450:50 человек в час.

3) Скорость обслуживания на 1 кассе: 60+5 человек в час.

Каждый магазин имеет 1 погрузочно-разгрузочный подъезд. Разгрузка занимает время 1 час. Раскладка товара занимает еще 2+1 час.

Дискрет времени моделирования - 1 час.

Требуется оценить следующие параметры за время 2 месяца:

1) рекомендуемый объем складов магазинов и страхового запаса.

2) график и объем поставок каждого вида товаров по сети;

3) рекомендуемое количество касс в каждом магазине.

Описание интерфейса и моделирования

Для моделирования работы супермаркетов и поставок товаров было написано небольшое WPF приложение на языке C#.

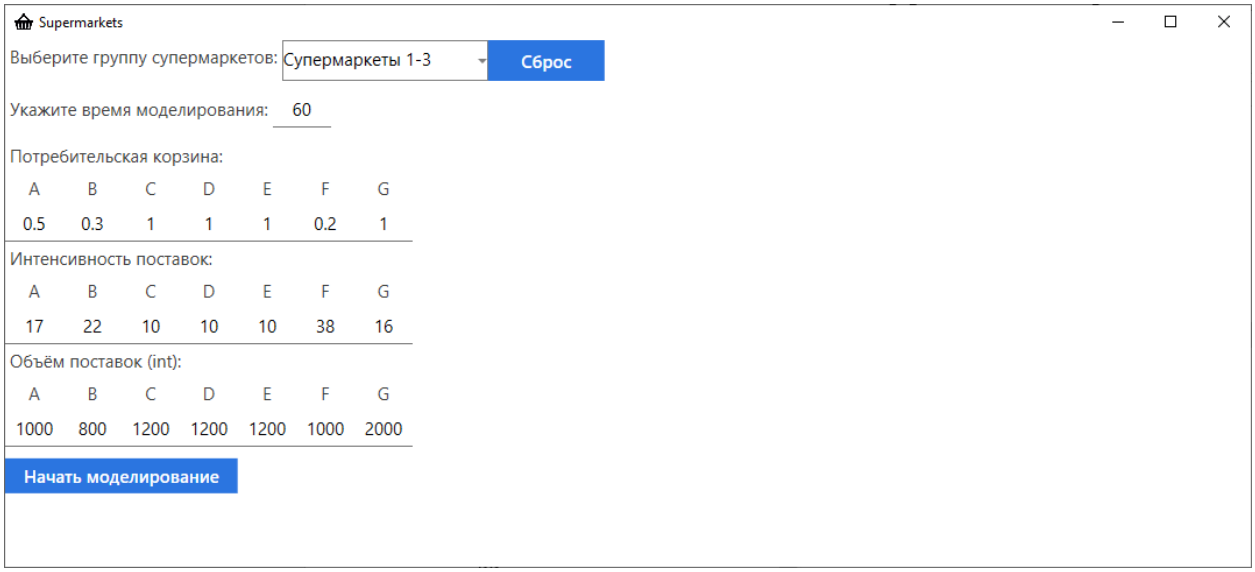

Рис 1. Интерфейс приложения

Поскольку супермаркеты 1-3, 4-7, 8-12 имеют одинаковые параметры – они были вынесены в группы и рассматривались как единое целое.

В качестве единиц модельного времени настроены часы. В стартовом окне с помощью выпадающего списка можно выбрать необходимую группу супермаркетов для моделирования, а также указать время моделирования. Ниже отображается потребительская корзина, интенсивность поставок и объем поставок для каждого супермаркета. По умолчанию используются параметры из задания и найденные нами значения для стабильной работы системы, но их также можно изменить, поменяв числа в соответствующих ячейках. Также здесь находятся кнопки «Сброс» и «Начать моделирование». «Сброс» - возвращает систему к исходным параметрам, а «Начать моделирование» - запускает моделирование системы.

Выберем первую группу супермаркетов и запустим моделирование.

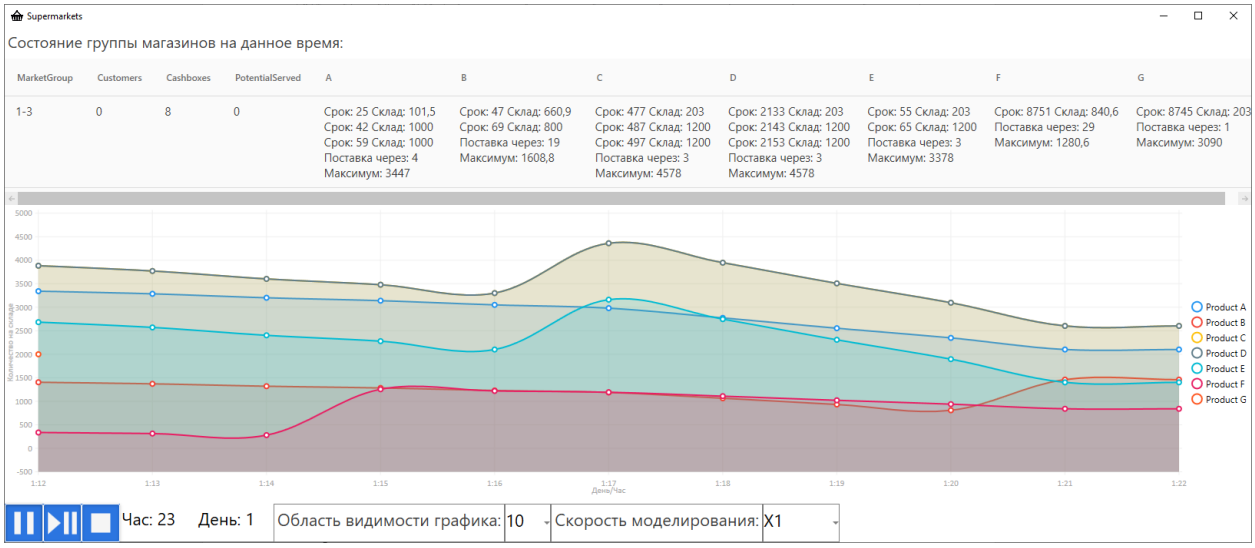

Рис 2. Состояние первой группы супермаркетов в первые дни работы работы

В верхней части окна находится таблица, показывающая состояние группы супермаркетов на данное время. Столбец MarketGroup отображает группу магазинов, Customers – количество посетителей в каждом магазине за последний час, Cashboxes – количество кассовых аппаратов, PotentialServed – максимальное количество посетителей, которое могли обслужить кассы за последний час (кол-во касс * (60 ± 5)). Столбцы A-G показывают состояние соответствующего вида товара на данный момент. Через заданные интервалы в магазин поступает заданный объём того или иного товара. Для каждой поставки отображается оставшееся время хранения в часах и объём. Также можно наблюдать за временем до следующей поставки и максимальным объёмом товара за всё время.

Ниже находится график, показывающий изменение общего количества каждого товара на складе в зависимости от времени. Внизу окна располагается панель управления. С помощью неё можно останавливать и возобновлять моделирования, управлять масштабом графика и скоростью моделирования, также здесь отображается текущее время

Выберем виртуальную скорость моделирования и сразу смоделируем 2 месяца работы.

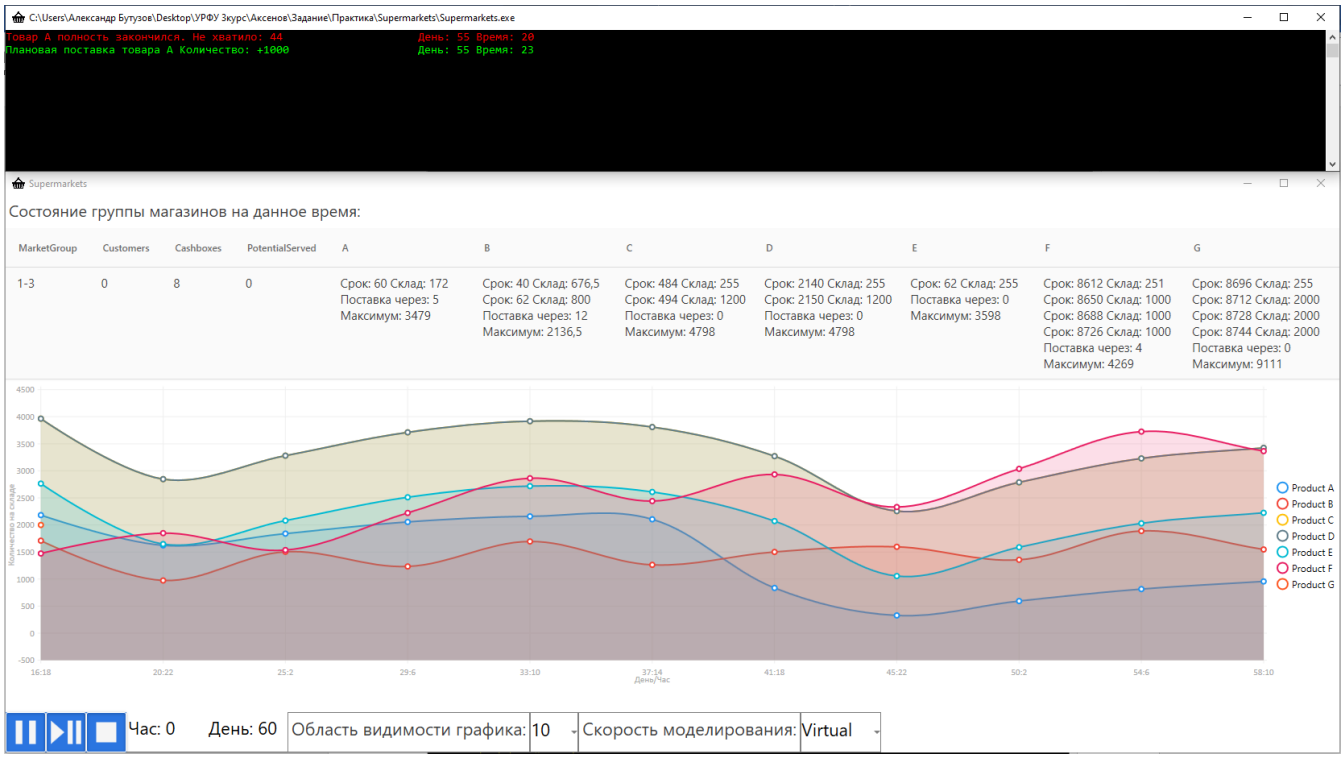

Рис 3. Результат моделирования первой группы магазинов за 2 месяца.

Как видно из сообщений – за 2 месяца работы только 1 раз возникла ситуация, когда какой-то товар полностью закончился, что является вполне неплохим результатом. Кроме того, ни один товар за всё время не был просрочен, вместе с тем ни один из продуктов долгого хранения не начал переполняться, что также является отличным результатом.

Для примера, изменим параметры и посмотрим, что будет, если одни товары доставлять слишком часто, а другие слишком редко.

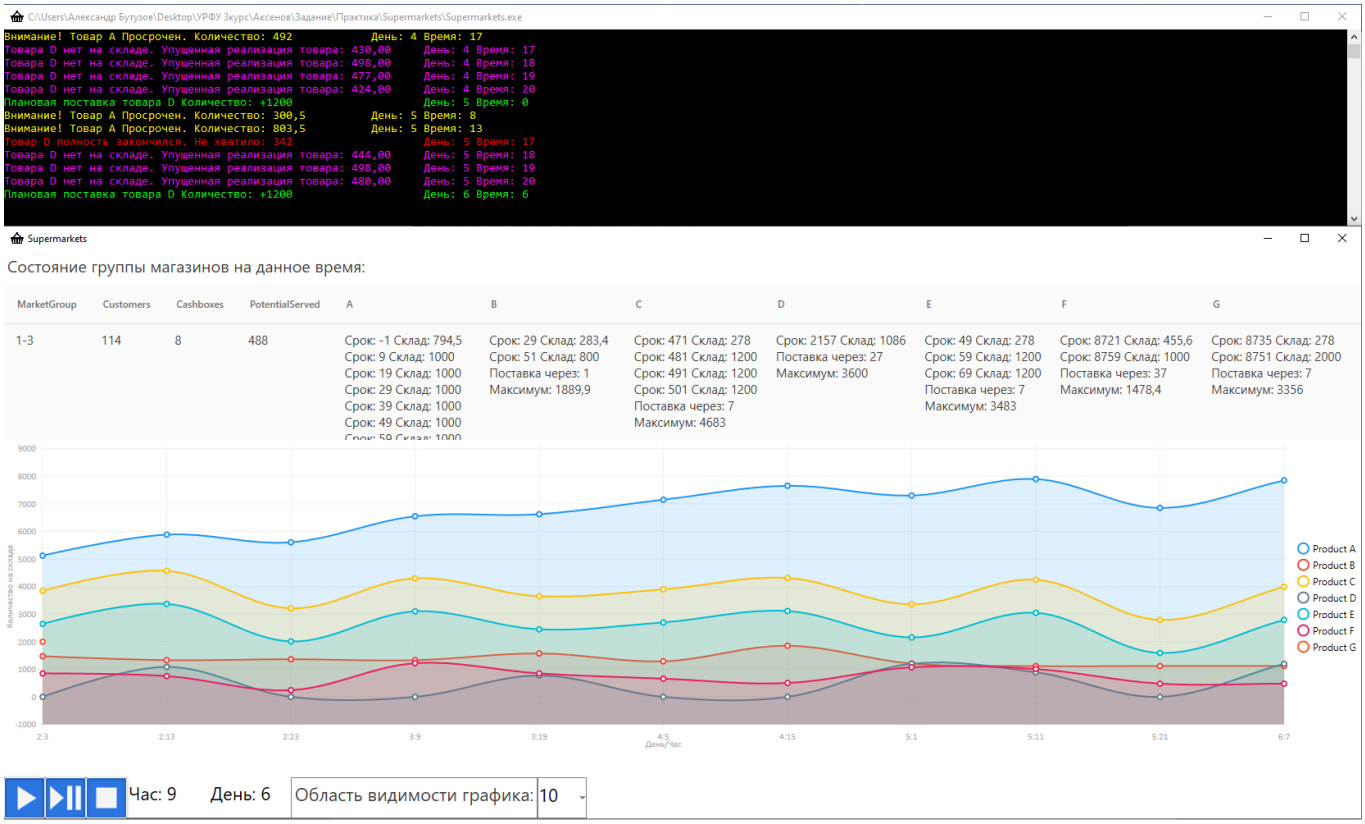

Рис 5. Состояние супермаркетов при некорректной настройке системы

Как видно из результатов, при такой настройке товар A не успевают продать, и он портится и переполняется, а товар D постоянно заканчивается,

Оценка параметров системы и ответы на вопросы

Исходя из результатов моделирования можно определить необходимые объёмы складов для каждой группы магазинов, а также необходимый страховой запас. Для этого воспользуемся параметром «Максимум» для каждого товара и посмотрим, какой максимальный объём данного товара был на складе за всё время моделирования:

Таблица 1

Максимальное количество товаров каждого вида на складе за 2 месяца

	A	B	C	D	E	F	G
Супермаркеты (1-3)	3504	2067	4701	4701	3727	4213	8907
Супермаркеты (4-7)	6920	7743	23648	11980	3430	4159	6976
Супермаркеты (8-12)	4056	5323	12072	8731	3634	8835	9994

В качестве страхового запаса для склада возьмём объём равный 1 поставке и получим следующие рекомендуемые объемы складов магазинов:

Таблица 2

Рекомендуемые объемы складов супермаркетов с учетом страхового запаса

	A	B	C	D	E	F	G
Супермаркеты (1-3)	4500	2800	6000	6000	5000	5300	11000
Супермаркеты (4-7)	10000	10000	27200	14400	4600	5200	8000
Супермаркеты (8-12)	6000	7700	16800	12400	4800	10800	12000

Также, в результате моделирования был определен следующий график и объем поставок каждого вида товаров по сети.

Таблица 3

График и объем поставок каждого вида товаров для 1-3 супермаркетов

	A	B	C	D	E	F	G
Первая поставка	3000	1600	3600	3600	2400	1000	2000
Интервал	17	22	10	10	10	38	16
Объём поставок	1000 (1 грузовик)	800 (1 грузовик)	1200 (1 грузовик)	1200 (1 грузовик)	1200 (1 грузовик)	1000 (1 грузовик)	2000 (2 грузовика)

Таблица 4

График и объем поставок каждого вида товаров для 4-7 супермаркетов

	A	B	C	D	E	F	G
Первая поставка	3000	3200	4800	2400	2400	1000	1000
Интервал	25	20	15	44	10	20	25
Объём поставок	3000 (3 грузовика)	2400 (3 грузовика)	3600 (3 грузовика)	2400 (2 грузовика)	1200 (1 грузовик)	1000 (1 грузовик)	1000 (1 грузовик)

Таблица 5

График и объем поставок каждого вида товаров для 8-12 супермаркетов

	A	B	C	D	E	F	G
Первая поставка	3000	3200	6000	4800	2400	2000	3000
Интервал	24	29	16	20	10	32	20
Объём поставок	2000 (2 грузовика)	2400 (3 грузовика)	4800 (4 грузовика)	3600 (3 грузовика)	1200 (1 грузовик)	2000 (2 грузовика)	2000 (2 грузовика)

Учитывая, что разгрузка товара занимает 2 ± 1 час – необходимо отправлять грузовики за 2-3 часа с интервалами в 1 час, если поставка состоит из нескольких грузовиков.

Кроме того, в ходе моделирования было определено, что в каждом супермаркете должно быть не меньше 8 касс, в таком случае почти всегда все покупатели, пришедшие в магазин, будут обслужены.

Заключение

В ходе производственной практики я получил большой опыт в сфере создания it-приложений и изучил основы компьютерного моделирования. Было создано приложение, которое позволяет имитировать работу сети супермаркетов, постоянно проверяя количество запасов всей имеющейся продукции при всех заданных условиях. За работой можно следить как с помощью табличных данных, так и с помощью графика. Так же скорость моделирования можно изменять в зависимости от предпочтений.

При написании приложения использовались следующие технологии:

- Паттерн MVVM, который нужен для разделения интерфейса и бизнес-логики;
- Библиотека LiveCharts для отображения графиков;
- Библиотека MaterialDesign для создания интерфейсов;
- Привязка XAML элементов к различным объектам и реагирование на их изменение (Binding);
- Асинхронное программирование для запуска течения времени в отдельном потоке

Использование новых библиотек и техник программирования помогло мне получить новые навыки в области программирования и проектирования приложений, а также улучшить уже существующие. Безусловно, эти полезные знания пригодятся в будущей профессии.

В приложении были реализованы все поставленные задачи, а также остались возможности для будущего развития и обновления проекта.

Ссылка на приложение:
https://drive.google.com/file/d/148_LIRqs1U_NM6lfUP2sMCnJeIqivezv/view?usp=sharing

Ссылка на репозиторий GitHub:
https://github.com/Lepricon74/Supermarkets

Список источников

1. https://www.cyberforum.ru/ - общие вопросы в сфере программирования.
2. https://lvcharts.net/ - руководство по использованию графиков.
3. https://metanit.com/sharp/wpf/22.1.php – руководство по использованию паттерна MVVM.
4. https://m.habr.com/ru/post/338518/ - руководство по использованию паттерна MVVM.
5. http://materialdesigninxaml.net/ - руководство по использованию библиотек для оформления интерфейсов.

Приложение А

1. Код программы на языке C#

```
using Prism.Mvvm;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Threading;
using LiveCharts;
using LiveCharts.Wpf;

namespace Supermarkets.Models
{
    class TimeMarketModel : BindableBase
    {
        public Random rnd = new Random();
        public bool pause;
        public int SelectedMarket = 0;
        public int DelayTime = 1;
        public int ModelingTime = 60;
        private int[] speeds = new int[] { 1000, 500, 200, 100, 20, 0};
        public int[] ChartSteps = new int[] { 1, 2, 5, 10, 50, 100};
        public int[] ChartDetails = new int[] { 10, 15, 20, 25, 30};
        public int CurrentSpeed = 0;
        public int ChartDetail = 0;
        private string[] productsNames = { "A", "B", "C", "D", "E", "F", "G" };
        private int[] productsTerm = { 72, 72, 504, 2160, 72, 8760, 8760 };

        public ObservableCollection<int> ProductsSupplies;
        public ObservableCollection<int> SupplyIntervals;
        public ObservableCollection<double> ShoppingCart;

        private int currentDay = 0;

        public int ChartStep = 1;
        public string CurrentDay { get { return "День: " + currentDay; } }

        private int currentTime = 0;
        public string CurrentTime{ get { return "Час: " + currentTime; } }

        public SeriesCollection Chart = new SeriesCollection();
        public List<string> Labels = new List<string>();
        private Market market;

        public ObservableCollection<Market> Market { get; }


        public TimeMarketModel()
        {
            Market = new ObservableCollection<Market>();
            ChangeMarket(0,true);
        }
        private void SetStartChart()
        {
            ProductList[] productSums = market.ToArray();
            for (int i = 0; i < 7; i++)
            {
                Chart.Add(new LineSeries
                {
                    Title = "Product " + productsNames[i],
                    Values = new ChartValues<double> { productSums[i].CurrentSum }
                });
            }
            Labels.Add(currentDay + ":" + currentTime);
        }

        public void OneHour()
        {
            int customersInHour = 0;
            market.Customers = 0;
            market.PotentialServed = 0;
```

```
                for (int i = 0; i < 7; i++)
                {
                    if (market.ToArray()[i].NextSupply == 0)
                    {
                        if (market.ToArray()[i].List.Count == 0)
                        {
                            Console.ForegroundColor = ConsoleColor.Green;
                            Console.WriteLine("Плановая поставка товара " + productsNames[i] + "
    Количество: +" + ProductsSupplies[i] + "\t\t\tДень: " + currentDay + " Время: " + currentTime);
                        }
                        market.ToArray()[i].Add(new Product(productsTerm[i], ProductsSupplies[i]));
                        market.ToArray()[i].UpdateMax();
                        market.ToArray()[i].NextSupply = SupplyIntervals[i];
                    }
                    market.ToArray()[i].NextSupply--;
                }

                if (currentTime >= 8 && currentTime < 21)
                {
                    if (currentTime >= 8 && currentTime < 13) customersInHour = rnd.Next(80, 120);
                    if (currentTime >= 13 && currentTime < 17) customersInHour = rnd.Next(120, 180);
                    if (currentTime >= 17 && currentTime < 21) customersInHour = rnd.Next(400, 500);
                    market.Customers = customersInHour;
                    market.PotentialServed = rnd.Next(55, 65) * market.Cashboxes;

                    for (int i = 0; i < 7; i++)
                    {
                        if (market.ToArray()[i].List.Count == 0)

                        {
                        Console.ForegroundColor = ConsoleColor.Magenta;
                        Console.WriteLine("Товара " + productsNames[i] + " нет на складе. " + "Упущенная
реализация товара: " + String.Format("{0:0.00}", ShoppingCart[i] * customersInHour) + "\tДень: " +
currentDay + " Время: " + currentTime);
                        continue;
                        }

                    Product product = market.ToArray()[i].List[0];
                    if (product.term < 0)
                    {
                        Console.ForegroundColor = ConsoleColor.Yellow;
                        Console.WriteLine("Внимание! Товар " + productsNames[i] + " Просрочен." + "
Количество: " + product.count + "\t\tДень: " + currentDay + " Время: " + currentTime);
                        market.ToArray()[i].List.RemoveAt(0);
                        continue;
                    }
                    if (market.ToArray()[i].List.Count == 1)
                    {
                        double productBought = ShoppingCart[i] * customersInHour;
                        if (product.count - productBought >= 0)
                        {
                            product.count = Math.Round(product.count - productBought, 2);
                            continue;
                        }
                        else
                        {
                            market.ToArray()[i].RemoveFirst();
                            Console.ForegroundColor = ConsoleColor.Red;
                            Console.WriteLine("Товар " + productsNames[i] + " полность закончился. Не
хватило: " + Math.Round(productBought - product.count) + "\t\tДень: " + currentDay + " Время: " +
currentTime);
                            continue;
                        }
                    }
                    product.count = Math.Round(product.count - ShoppingCart[i] * customersInHour, 2);
                    if (product.count <= 0)
                    {
                        market.ToArray()[i].List[1].count += product.count;
                        market.ToArray()[i].List.RemoveAt(0);
                        continue;
                    }
                }
            }

            currentTime++;
```

```
            foreach (ProductList listProduct in market)
            {
                listProduct.UpdateCurrentSum();
                foreach (Product product in listProduct.List)
                {
                    product.term--;
                }
            }

            if (currentTime == 24)
            {
                currentDay++;
                RaisePropertyChanged("CurrentDay");
                currentTime = 0;
            }
            RaisePropertyChanged("CurrentTime");

        }


        public void ChangeMarket(int index, bool reset)
        {
            currentDay = 0;
            currentTime = 0;
            pause = true;
            SelectedMarket = index;

            if (index == 0)
            {
                if (reset)
                {
                    ShoppingCart = new ObservableCollection<double> { 0.5, 0.3, 1, 1, 1, 0.2, 1 };
                    SupplyIntervals = new ObservableCollection<int> { 17, 22, 10, 10, 10, 38, 16 };
                    ProductsSupplies = new ObservableCollection<int> { 1000, 800, 1200, 1200, 1200, 1000,
2000 };
                }
                market = new Market("1-3", 0, 8, 0, new Product(72, 3000), new Product(72, 1600), new
Product(504, 3600), new Product(2160, 3600), new Product(72, 2400), new Product(8760, 1000), new
Product(8760, 2000), SupplyIntervals);
            }
            if (index == 1)
            {
                if (reset)
                {
                    ShoppingCart = new ObservableCollection<double> { 1, 1, 1.9, 0.4, 1, 0.4, 0.3 };
                    SupplyIntervals = new ObservableCollection<int> { 25, 20, 15, 44, 10, 20, 25 };
                    ProductsSupplies = new ObservableCollection<int> { 3000, 2400, 3600, 2400, 1200, 1000,
1000 };
                }
                market = new Market("4-7", 0, 8, 0, new Product(72, 3000), new Product(72, 3200), new
Product(504, 4800), new Product(2160, 2400), new Product(72, 2400), new Product(8760, 1000), new
Product(8760, 1000), SupplyIntervals);
            }
            if (index == 2)
            {
                if (reset)
                {
                    ShoppingCart = new ObservableCollection<double> { 0.7, 0.7, 2.5, 1.5, 1, 0.5, 0.8 };
                    SupplyIntervals = new ObservableCollection<int> { 24, 28, 16, 20, 10, 32, 20 };
                    ProductsSupplies = new ObservableCollection<int> { 2000, 2400, 4800, 3600, 1200, 2000,
2000 };
                }
                market = new Market("8-12", 0, 8, 0, new Product(72, 5000), new Product(72, 3200), new
Product(504, 6000), new Product(2160, 4800), new Product(72, 2400), new Product(8760, 2000), new
Product(8760, 3000), SupplyIntervals);
                }

            Labels.Clear();
            Chart.Clear();
            Market.Clear();
            Market.Add(market);
            ChartDetail = 0;
            SetStartChart();
            RaisePropertyChanged("CurrentTime");
            RaisePropertyChanged("CurrentDay");
            Console.Clear();
```

```
            if (reset)
            {
                RaisePropertyChanged("ShoppingCart");
                RaisePropertyChanged("SupplyIntervals");
                RaisePropertyChanged("ProductsSupplies");
            }
        }

        public void UpdateSupplyParametersInProducts()
        {

            for (int i = 0; i < 7; i++)
            {
                    market.ToArray()[i].NextSupply = SupplyIntervals[i];
            }
        }

        public void OneHourSync()
        {
            OneHour();
            UpdateIntetface();
        }
        public async void OneHourAsync()
        {
            while (currentDay < ModelingTime)
            {
                if (pause) return;
                else
                {
                    await Task.Run(() => OneHour());
                    await Task.Delay(speeds[CurrentSpeed]);
                    UpdateIntetface();
                }

            }
        }

        public void UpdateIntetface()
        {
            ChartStep--;
            Market.Clear();
            Market.Add(market);
            if (ChartStep == 0)
            {
                while (Labels.Count > ChartDetails[ChartDetail]) Labels.RemoveAt(0);
                Labels.Add(currentDay + ":" + currentTime);
                ProductList[] productSums = market.ToArray();
                for (int i = 0; i < 6; i++)
                {
                    while (Chart[i].Values.Count > ChartDetails[ChartDetail]) Chart[i].Values.RemoveAt(0);
                    Chart[i].Values.Add(productSums[i].CurrentSum);
                }
                ChartStep = ChartSteps[CurrentSpeed];
            }
        }
    }
}
```

Приложение В

1. Код класса Market

```
namespace Supermarkets
{
    class Market : BindableBase, IEnumerable
    {
        public Market(string group, int _customers, int _cashboxes, int _served, Product a, Product b,
Product c, Product d, Product e, Product f, Product g, ObservableCollection<int> _supplyIntervals)
        {
            this.MarketGroup = group;
            this.Customers = _customers;
            this.Cashboxes = _cashboxes;
            this.PotentialServed = _served;
            this.A = new ProductList(a, _supplyIntervals[0]);
            this.B = new ProductList(b, _supplyIntervals[1]);
            this.C = new ProductList(c, _supplyIntervals[2]);
            this.D = new ProductList(d, _supplyIntervals[3]);
            this.E = new ProductList(e, _supplyIntervals[4]);
            this.F = new ProductList(f, _supplyIntervals[5]);
            this.G = new ProductList(g, _supplyIntervals[6]);
        }
        public string MarketGroup { get; set; }
        public int Customers { get; set; }
        public int Cashboxes { get; set; }
        public int PotentialServed { get; set; }
        public ProductList A { get; set; }
        public ProductList B { get; set; }
        public ProductList C { get; set; }
        public ProductList D { get; set; }
        public ProductList E { get; set; }
        public ProductList F { get; set; }
        public ProductList G { get; set; }

        public ProductList[] ToArray()
        {
            return new ProductList[] { this.A, this.B, this.C, this.D, this.E, this.F, this.G };
        }

        public IEnumerator GetEnumerator()
        {
            return ToArray().GetEnumerator();
        }
    }
}
```

2. Код класса ProductList

```
namespace Supermarkets
{
    class ProductList
    {
        public int NextSupply { get; set; }
        public double MaxCount { get; set; }

        public double CurrentSum;
        public List<Product> List { get; set; }

        public ProductList(Product product, int _nextSupply)
        {
            this.List = new List<Product>() { product };
            this.NextSupply = _nextSupply;
            this.MaxCount = product.count;
            this.CurrentSum = product.count;
        }
        public void RemoveFirst()
        {
            this.List.RemoveAt(0);
        }
        public void Add(Product product)
        {
            this.List.Add(product);
```

```
        }
        public void UpdateMax()
        {
            UpdateCurrentSum();
            if (CurrentSum > this.MaxCount) this.MaxCount = CurrentSum;

        }

        public void UpdateCurrentSum()
        {
            CurrentSum = 0;
            foreach (Product product in this.List)
            {
                CurrentSum += product.count;
            }
        }
        public override string ToString()
        {
            string result = "";
            foreach (Product product in this.List)
            {
                result = result + product.ToString() + "\n";
            }
            result = result + "Поставка через: " + this.NextSupply + "\n";
            result = result + "Максимум: " + this.MaxCount;
            return result;
        }
    }
}
```

3. Код класса Product

```
namespace Supermarkets
{
    class Product
    {
        public int term;
        public double count;

        public Product(int date, double _quantity)
        {
            this.term = date;
            this.count = _quantity;
        }
        public override string ToString()
        {

            return ("Срок: " + this.term.ToString() + " " + "Склад: " + this.count.ToString());
        }
    }
}
```

Приложение С

1. Код класса TimeMarketViewModel

```
class TimeMarketViewModel : BindableBase
{
    readonly TimeMarketModel _model = new TimeMarketModel();
    public ObservableCollection<Market> Markets => _model.Market;

    public SeriesCollection Chart => _model.Chart;
    public List<String> Labels => _model.Labels;
    public TimeMarketViewModel()
    {
        _model.PropertyChanged += (s, e) => { RaisePropertyChanged(e.PropertyName); };
        StartPause = new DelegateCommand(() => {
            if (_model.pause)
            {
                _model.pause = false;
                _model.OneHourAsync();
            }
            else
            {
                _model.pause = true;
            }

        });
        Stop = new DelegateCommand(() => {
            _model.ChangeMarket(_model.SelectedMarket, false);
            _model.CurrentSpeed = 0;

        });
        Reset = new DelegateCommand(() => { _model.ChangeMarket(_model.SelectedMarket, true); });
        Step = new DelegateCommand(() => { _model.OneHourSync(); });
        UpdateSupply = new DelegateCommand(() => { _model.UpdateSupplyParametersInProducts(); });
    }

    public int ModelingTime
    {
        get { return _model.ModelingTime; }
        set
        {
            if ((value > 360) || (value < 1)) _model.ModelingTime = 60;
            else _model.ModelingTime = value;
            RaisePropertyChanged("ModelingTime");
        }
    }
    public string CurrentTime => _model.CurrentTime;
    public string CurrentDay => _model.CurrentDay;
    public ObservableCollection<int> SupplyIntervals
    {
        get { return _model.SupplyIntervals; }
        set
        {

            _model.SupplyIntervals = value;
            RaisePropertyChanged("SupplyIntervals");
        }
    }
    public ObservableCollection<int> ProductsSupplies
    {
        get { return _model.ProductsSupplies; }
        set
        {
            _model.ProductsSupplies = value;
            RaisePropertyChanged("SupplyIntervals");
        }
    }
    public ObservableCollection<double> ShoppingCart
    {
        get { return _model.ShoppingCart; }
        set
        {
            _model.ShoppingCart = value;
            RaisePropertyChanged("ShoppingCart");
        }
    }
```

```
        public int SelectMarket
        {
            get { return _model.SelectedMarket; }
            set
            {
                _model.ChangeMarket(value,true);

            }
        }
        public int SelectChartDetail
        {
            get { return _model.ChartDetail; }
            set
            {
                _model.ChartDetail=value;

            }
        }
        public int SelectSpeed
        {
            get { return _model.CurrentSpeed; }
            set
            {
                _model.CurrentSpeed = value;
                _model.ChartStep = _model.ChartSteps[value];
            }
        }
        public DelegateCommand StartPause { get; }
        public DelegateCommand Stop { get; }
        public DelegateCommand Reset { get; }
        public DelegateCommand Step { get; }

        public DelegateCommand UpdateSupply  {get; }
    }
```

Приложение D

1. Разметка XAML

```
<Window x:Class="Supermarkets.MainWindow"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:lvc= "clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
        xmlns:local="clr-namespace:MyMarkets"
        mc:Ignorable="d"
        Title="Supermarkets" Height="730" Width="1672">

    <Grid>
        <StackPanel x:Name="SettingsWindow" Visibility="Visible">
            <Grid >
                <StackPanel Orientation="Horizontal">
                    <Label Content="Выберите группу супермаркетов:" FontSize="15"/>
                    <ComboBox  BorderBrush="Gray" BorderThickness="1"  HorizontalAlignment="Left"
VerticalAlignment="Top" Width="177" Foreground="Black" Height="35"  StaysOpenOnEdit="True"
IsEditable="True" FontSize="15" SelectedIndex="{Binding SelectMarket}">
                        <ComboBoxItem Content="Супермаркеты 1-3"/>
                        <ComboBoxItem Content="Супермаркеты 4-7"/>
                        <ComboBoxItem Content="Супермаркеты 8-12"/>
                    </ComboBox>
                    <Grid Background="#FF2B75E0">
                        <Button FontSize="15" Width="100" Height="30" HorizontalAlignment="Left"
Content="Сброс" Command="{Binding Reset}" Background="{x:Null}" Foreground="White"
BorderBrush="{x:Null}"></Button>
                    </Grid>
                </StackPanel>
            </Grid>
            <StackPanel Orientation="Horizontal" Margin="0,10,0,10">
                <Label Content="Укажите время моделирования:" FontSize="15" />
                <TextBox HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Text="{Binding ModelingTime, Mode=TwoWay}" Width="50" FontSize="15">
                    <TextBox.ToolTip>
                        Введите время моделирования в днях
                    </TextBox.ToolTip>
                </TextBox>
            </StackPanel>
            <Grid >
                <Label Content="Потребительская корзина:" FontSize="15" />
            </Grid>
            <Grid >
                <StackPanel Orientation="Horizontal">
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="A" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="B" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="C" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="D" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="E" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="F" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="G" Width="50" FontSize="15"/>
                </StackPanel>
            </Grid>
            <StackPanel Orientation="Horizontal">
            <Grid>
                    <Grid.ToolTip>
                        Введите число
                    </Grid.ToolTip>
                <StackPanel  x:Name="CartOutput" Orientation="Horizontal"  DataContext="{Binding
ShoppingCart,Mode=TwoWay}" >
                    <TextBox x:Name="ACart" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [0]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="BCart" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [1]}" Width="50" FontSize="15"/>
```

```
                    <TextBox x:Name="CCart" HorizontalContentAlignment="Center"
    HorizontalAlignment="Left" Height="30"  Text="{Binding [2]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="DCart" HorizontalContentAlignment="Center"
    HorizontalAlignment="Left" Height="30"  Text="{Binding [3]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="ECart" HorizontalContentAlignment="Center"
    HorizontalAlignment="Left" Height="30"  Text="{Binding [4]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="FCart" HorizontalContentAlignment="Center"
    HorizontalAlignment="Left" Height="30"  Text="{Binding [5]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="GCart" HorizontalContentAlignment="Center"
    HorizontalAlignment="Left" Height="30"  Text="{Binding [6]}" Width="50" FontSize="15"/>

                </StackPanel>
            </Grid>
            </StackPanel>
            <Grid >
                <Label Content="Интенсивность поставок:" FontSize="15" />
            </Grid>
            <Grid >
                <StackPanel Orientation="Horizontal">
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="A" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="B" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="C" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="D" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="E" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="F" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
    Content="G" Width="50" FontSize="15"/>
                </StackPanel>
            </Grid>
            <StackPanel Orientation="Horizontal">
            <Grid>
                <Grid.ToolTip>
                    Введите целое число
                </Grid.ToolTip>
                <StackPanel  DataContext="{Binding SupplyIntervals,Mode=TwoWay}"
    Orientation="Horizontal">
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [0]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [1]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [2]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [3]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [4]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [5]}" Width="50" FontSize="15"/>
                    <TextBox  HorizontalContentAlignment="Center" HorizontalAlignment="Left"
    Height="30"  Text="{Binding [6]}" Width="50" FontSize="15"/>
                </StackPanel>
            </Grid>
            </StackPanel>
            <Grid >
                <Label Content="Объём поставок (int):" FontSize="15" />
            </Grid>
            <Grid >

<StackPanel Orientation="Horizontal">
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="A" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="B" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="C" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="D" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="E" Width="50" FontSize="15"/>
```

```
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="F" Width="50" FontSize="15"/>
                    <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="30"
Content="G" Width="50" FontSize="15"/>
                </StackPanel>
            </Grid>
            <StackPanel Orientation="Horizontal">
            <Grid >
                    <Grid.ToolTip>
                        Введите целое число
                    </Grid.ToolTip>
                <StackPanel   Orientation="Horizontal" DataContext="{Binding
ProductsSupplies,Mode=TwoWay}" >
                    <TextBox x:Name="ASupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [0]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="BSupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [1]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="CSupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [2]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="DSupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [3]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="ESupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [4]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="FSupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [5]}" Width="50" FontSize="15"/>
                    <TextBox x:Name="GSupplyCount" HorizontalContentAlignment="Center"
HorizontalAlignment="Left" Height="30"  Text="{Binding [6]}" Width="50" FontSize="15"/>
                </StackPanel>
            </Grid>
            </StackPanel>

            <StackPanel Orientation="Horizontal">
            <Grid Margin="0,10,0,0" Background="#FF2B75E0" Width="Auto">
                    <Button x:Name="Modeling" FontSize="15" Width="Auto" Height="30"
HorizontalAlignment="Left" Click="Modeling_Click" Background="{x:Null}" Command="{Binding UpdateSupply}"
Foreground="White" BorderBrush="{x:Null}">Начать моделирование</Button>
            </Grid>
            </StackPanel>
        </StackPanel>
        <Grid x:Name="ModelWindow" Visibility="Collapsed" >
        <StackPanel >
            <Grid >
                <Label  HorizontalContentAlignment="Center" HorizontalAlignment="Left" Height="40"
Content="Состояние группы магазинов на данное время:"  FontSize="20"/>
            </Grid>
            <DataGrid ItemsSource="{Binding Markets}"  Height="190" FontSize="15"></DataGrid>
            <lvc:CartesianChart Series="{Binding Chart}"  LegendLocation="Right" Height="384" >
                <lvc:CartesianChart.AxisY>
                    <lvc:Axis Title="Количество на складе" Labels="{Binding Chart}" ></lvc:Axis>
                </lvc:CartesianChart.AxisY>
                <lvc:CartesianChart.AxisX>
                    <lvc:Axis Title="День/Час" Labels="{Binding Labels}" ></lvc:Axis>
                </lvc:CartesianChart.AxisX>
            </lvc:CartesianChart>

        </StackPanel>
        <StackPanel Orientation="Horizontal" VerticalAlignment="Bottom">
            <Grid  Background="#FF2B75E0">
                <materialDesign:PackIcon x:Name="PlayPause"  Foreground="WhiteSmoke" Kind="Play"
Width="50" Height="50" HorizontalAlignment="Left" VerticalAlignment="Center" />
                <Button x:Name="StartModeling"  Width="50" Height="50" HorizontalAlignment="Left"
Background="{x:Null}" BorderBrush="White" Command="{Binding StartPause}" Click="StartModeling_Click"/>
            </Grid>
            <Grid  Background="#FF2B75E0">
                <materialDesign:PackIcon Foreground="WhiteSmoke" Kind="PlayPause" Width="50" Height="50"
HorizontalAlignment="Left" VerticalAlignment="Center" />
                <Button   FontSize="15" Background="{x:Null}" BorderBrush="White" Width="50" Height="50"
HorizontalAlignment="Left" Command="{Binding Step}"  />
            </Grid>
            <Grid  Background="#FF2B75E0">
                <materialDesign:PackIcon Foreground="WhiteSmoke" Kind="Stop" Width="50" Height="50"
HorizontalAlignment="Left" VerticalAlignment="Center" />
                <Button x:Name="StopModeling" Background="{x:Null}" BorderBrush="White" FontSize="15"
Width="50" Height="50" HorizontalAlignment="Left" Command="{Binding Stop}" Click="StopModeling_Click" />
            </Grid>
            <TextBox  HorizontalAlignment="Left" Height="50"  Text="{Binding CurrentTime, Mode=OneWay}"
Width="100" FontSize="22"/>
```

```
            <TextBox  HorizontalAlignment="Left" Height="50"  Text="{Binding CurrentDay, Mode=OneWay}"
Width="100" FontSize="22"/>
            <Label FontSize="22" BorderBrush="Gray" BorderThickness="1" Content="Область видимости
графика:"/>
            <ComboBox  BorderBrush="Gray" BorderThickness="1"  HorizontalAlignment="Left"
VerticalAlignment="Top" Width="60" Foreground="Black" Height="50"  StaysOpenOnEdit="True"
IsEditable="True" FontSize="22" SelectedIndex="{Binding SelectChartDetail}">
                <ComboBoxItem Content="10"/>
                <ComboBoxItem Content="15"/>
                <ComboBoxItem Content="20"/>
                <ComboBoxItem Content="25"/>
                <ComboBoxItem Content="30"/>
            </ComboBox>
            <StackPanel x:Name="SpeedsButtons" Orientation="Horizontal" Visibility="Collapsed">
                <Label FontSize="22" BorderBrush="Gray" BorderThickness="1" Content="Скорость
моделирования:"/>
                    <ComboBox x:Name="SpeedBox"   BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Foreground="Black" Height="50"
StaysOpenOnEdit="True" IsEditable="True" FontSize="22" SelectedIndex="{Binding SelectSpeed}">
                    <ComboBoxItem Content="X1"/>
                    <ComboBoxItem Content="X2"/>
                    <ComboBoxItem Content="X5"/>
                    <ComboBoxItem Content="X10"/>
                    <ComboBoxItem Content="X50"/>
                    <ComboBoxItem Content="Virtual"/>
                </ComboBox>

            </StackPanel>
        </StackPanel>
        </Grid>
    </Grid>
</Window>
```