МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Web-технологии»
Тема: Модуль администрирования приложения «Биржа акций»

Студент гр.8303 _____ Рудько Д.Ю.
Преподаватель _____ Беляев С.А.

Санкт-Петербург
2020

Цель работы.

Изучить основы языка TypeScript и особенностей применения фреймворка Angular для разработки web-приложений.

Задание.

Для достижения поставленной цели требуется решить следующие **задачи:**

1) Разработка интерфейса web-приложения.

2) Создание web-сервера на основе express, настройка маршрутов, подготовка и обработка REST-запросов с учетом CORS (серверная часть).

3) Создание каркаса web-приложения с использованием Angular.

4) Определение перечня компонентов и сервисов web-приложения.

5) Создание шаблонов компонентов.

6) Обеспечение взаимодействия с сервером приложения.

Необходимо создать web-приложение, обеспечивающее настройку биржи брокера, в которой есть возможность задать перечень участников, перечень акций, правила изменения акций во времени, время начала и время окончания торгов.

Основные требования:

1. Информация о брокерах (участниках) и параметрах акций сохраняется в файле в формате JSON.

2. В качестве сервера используется Node.JS с модулем express.

3. Предусмотрена HTML-страница с перечнем потенциальных брокеров. Брокеров можно добавлять и удалять, можно изменить начальный объём денежных средств.

4. Предусмотрена HTML-страница для перечня акций. Для каждой акции задаются правила изменения во времени (закон распределения: равномерный, нормальный; максимальное значение для изменения, общее количество доступных акций, начальная стоимость одной акции). Предусмотрена возможность добавления и удаления акций.

5. Предусмотрена HTML-страница для настроек биржи (время начала и окончания торгов, интервал времени, через который пересчитывается стоимость акций).

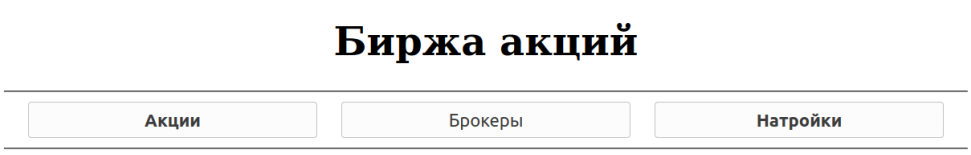
6. Все элементы управления реализованы с использованием компонентов Angular. Взаимодействие между компонентами реализовано с использованием сервисов Angular.

7. Для реализации эффектов на HTML-страницах используются директивы Angular.

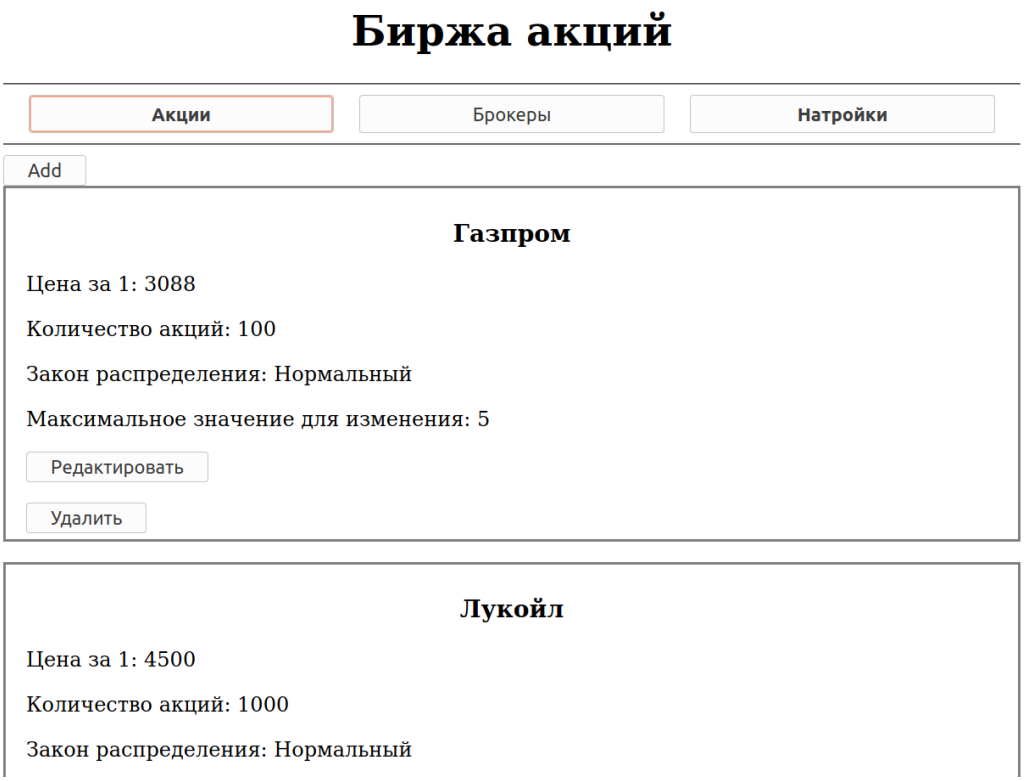
Ход выполнения.

Создан web-сервер на основе express. Созданы шаблоны компонент участников, акций и настроек, реализован ввод из json файла. На сервере обработаны запросы на сохранение и загрузку данных. Создан каркас web-приложения с использованием Angular.

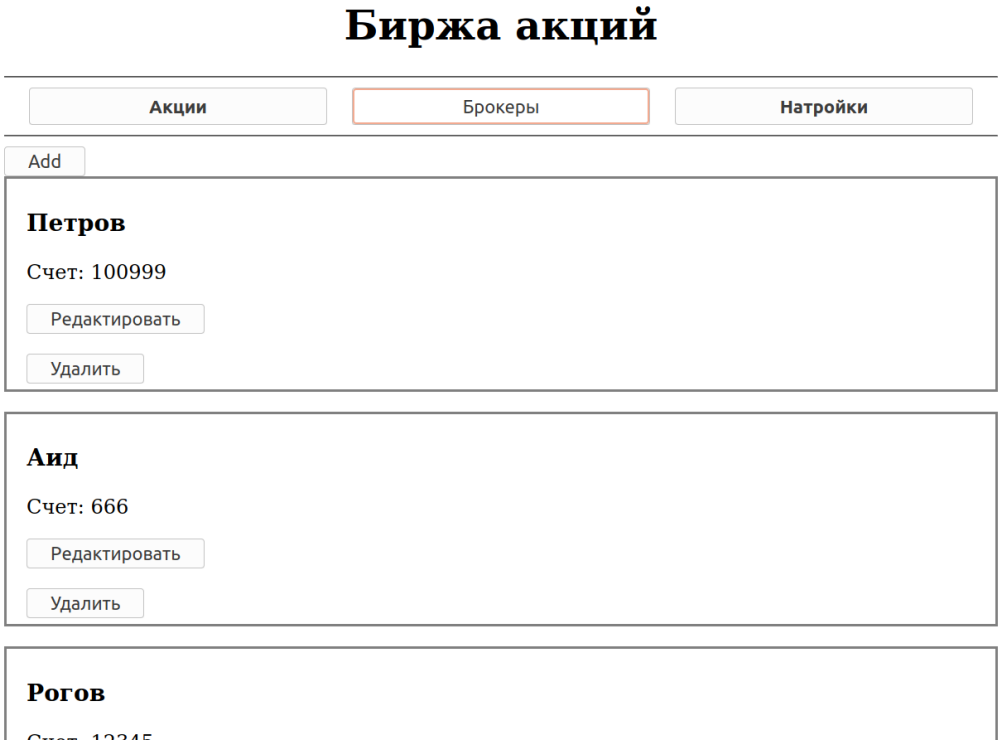
Главная страница



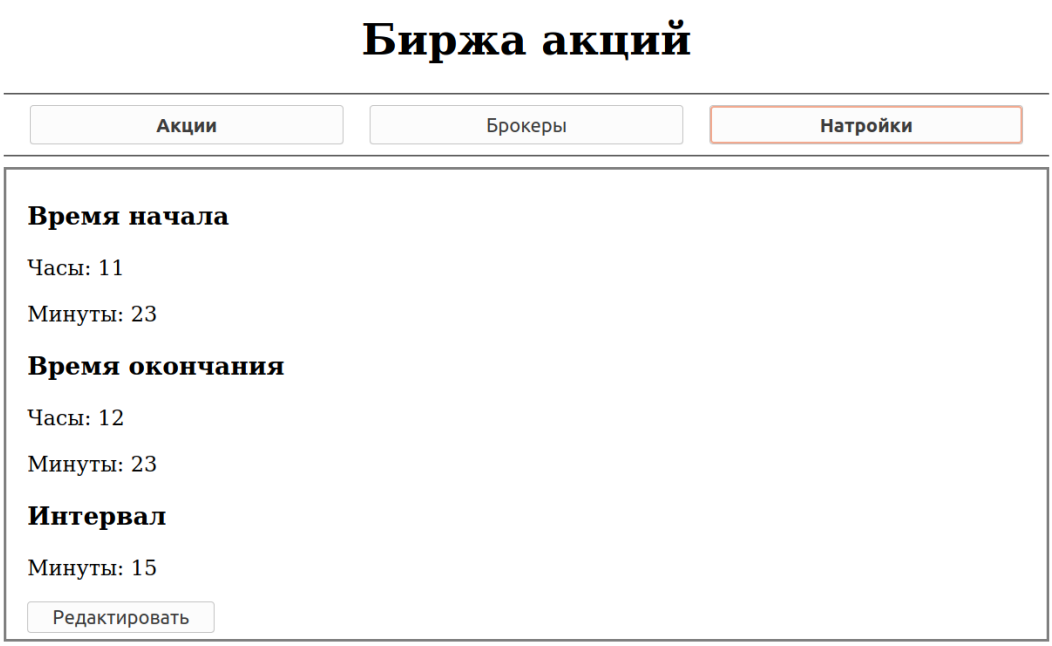
Страница акций



Страница брокеров



Страница настроек



Выводы.

В ходе выполнения лабораторной работы были изучены основы языка TypeScript и особенности применения фреймворка Angular для разработки web-приложений. Реализован модуль администрирования приложения «Биржи акций».

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

Сервер.

app.js

```
const express = require("express");
const fs = require('fs');
const path = require('path');
const app = express();

// создаем парсер для данных в формате json
const jsonParser = express.json();

// настройка CORS
app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
  res.header("Access-Control-Allow-Methods", "GET, PATCH, PUT, POST, DELETE,
OPTIONS");
  next();
});

var stocks = require('./stocks.json');
var brokers = require('./brokers.json');
var settings = require("./settings.json");

// обработчик по маршруту localhost:3000/postuser
app.post("/postuser", jsonParser, function (req, res) {

  // если не переданы данные, возвращаем ошибку
  if(!req.body) return res.sendStatus(400);

  if(req.body.name === 'Add'){
    //console.log('TUT');
    let newStock = {
      "title": req.body.title,
      "price": req.body.price,
      "count": req.body.count,
      "maxCount": req.body.maxCount
    }
    stocks.arr.push(newStock);

fs.writeFile(path.join(__dirname,'.','stocks.json'),JSON.stringify(stocks),funct
ion(err){
      if(err) throw err;
    })
    res.json(stocks);
  }

  if(req.body.name === 'Del'){
    var index = -1;
    for(let i = 0; i < stocks.arr.length; i++) {
      if (stocks.arr[i].title === req.body.title) {
        index = i;
      }
    }
    if (index !== -1) {
      stocks.arr.splice(index,1);
    }

fs.writeFile(path.join(__dirname,'.','stocks.json'),JSON.stringify(stocks),funct
ion(err){
      if(err) throw err;
    })
  }

  if(req.body.name === 'Change'){
    for(let i = 0; i < stocks.arr.length; i++){
      if(req.body.title === stocks.arr[i].title){
        stocks.arr[i].price = req.body.price;
        stocks.arr[i].count = req.body.count;
        stocks.arr[i].maxCount = req.body.maxCount;
```

```
      }
    }

fs.writeFile(path.join(__dirname,'.','stocks.json'),JSON.stringify(stocks),funct
ion(err){
      if(err) throw err;
    })

  }
  if(req.body.name === 'Get'){
    res.json(stocks);
  }
});

app.post("/broker", jsonParser, function (req, res){
  // если не переданы данные, возвращаем ошибку
  if(!req.body) return res.sendStatus(400);

  if(req.body.name === 'Get'){
    res.json(brokers);
  }

  if(req.body.name === 'Add'){
    let newBroker = {
      "brokerName" : req.body.brokerName,
      "money" : req.body.money
    }
    brokers.arr.push(newBroker)

fs.writeFile(path.join(__dirname,'.','brokers.json'),JSON.stringify(brokers),fun
ction(err){
      if(err) throw err;
    })
    res.json(brokers);
  }

  if(req.body.name === 'Change'){

    for(let i = 0; i < brokers.arr.length; i++){
      if(req.body.brokerName === brokers.arr[i].brokerName){
        brokers.arr[i].money = req.body.money;
      }
    }


fs.writeFile(path.join(__dirname,'.','brokers.json'),JSON.stringify(brokers),fun
ction(err){
      if(err) throw err;
    })
  }

  if(req.body.name === 'Del'){
    var index = -1;
    for(let i = 0; i < brokers.arr.length; i++) {
      if (brokers.arr[i].brokerName === req.body.brokerName) {
        index = i;
      }
    }
    if (index !== -1) {
      brokers.arr.splice(index,1);
    }


fs.writeFile(path.join(__dirname,'.','brokers.json'),JSON.stringify(brokers),fun
ction(err){
      if(err) throw err;
    })
  }

});

app.post("/set", jsonParser, function (req, res){
```

```
  if(!req.body) return res.sendStatus(400);
  if(req.body.name === 'Get'){
    res.json(settings);
  }
  if(req.body.name === 'Change'){
    settings.arr[0].beginTimeH = req.body.beginTimeH;
    settings.arr[0].beginTimeM = req.body.beginTimeM;
    settings.arr[0].endTimeH = req.body.endTimeH;
    settings.arr[0].endTimeM = req.body.endTimeM;
    settings.arr[0].interval = req.body.interval;

fs.writeFile(path.join(__dirname,'.','settings.json'),JSON.stringify(settings),f
unction(err){
      if(err) throw err;
    })
  }
})

app.listen(3000);
```

Клиент (Angular).

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import {HttpClientModule} from '@angular/common/http';
import {FormsModule} from '@angular/forms';

import { AppComponent } from './app.component';
import { StockComponent } from './stock/stock.component';
import { BrokerComponent } from './broker/broker.component';
import { SettingComponent } from './setting/setting.component';
import {BoldDirective} from '../bold.directive';

@NgModule({
    declarations: [
        AppComponent,
        StockComponent,
        BrokerComponent,
        SettingComponent,
        BoldDirective
    ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts

```
import { Component } from '@angular/core';
import  {HttpService} from '../http.service';

export interface Stock{
  arr: Stock[];
  title: string;
  price: string;
  count: string;
  maxCount: string;
}

export interface Broker{
  arr: Broker[];
  brokerName: string;
  money: string;
}

export interface Settings{
  arr: Settings[];
```

```
  beginTimeH: string;
  beginTimeM: string;
  endTimeH: string;
  endTimeM: string;
  interval: string;
}

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [HttpService]
})

export class AppComponent {
  title = 'Lab5';

  constructor(private httpService: HttpService){}

  private data: { price: string; name: string; count: string; title: string;
maxCount: string };
  private dataB: { money: string; name: string; brokerName: string };

  stocks: Stock[] = [];
  brokers: Broker[] = [];
  settings: Settings[] = [];

  done = false;
  doneB = false;
  doneS = false;
  visible = false;
  visibleB = false;
  visibleS = false;

  company = '';
  count = '';
  price = '';
  maxCount = '';
  brokerName = '';
  money = '';

  open = false;
  openBroker = false;
  openSettings = false;


  // Акции

  OpenStock(){
    if (this.open === false)  {
      this.submit('Get');
    }
    this.openBroker = false;
    this.openSettings = false;
    this.open = !this.open;
  }

  OpenAddStock(){
    this.visible = true;
  }

  Close(){
    this.visible = false;
    this.company = '';
    this.price = '';
    this.count = '';
    this.maxCount = '';
  }

  CloseAddStock(){
    this.visible = false;
    if( this.company !== '' &&
```

```
      this.price !== '' &&
      this.count !== '' &&
      this.maxCount !== ''){

    this.data = {
      name: 'Add',
      title: this.company,
      price: this.price,
      count: this.count,
      maxCount: this.maxCount
    };
    this.company = '';
    this.price = '';
    this.count = '';
    this.maxCount = '';
    this.AddStock(this.data);
  }
  else {
    alert('Вы заполнели не все поля. Акции компании не были добавлены');
    this.company = '';
    this.price = '';
    this.count = '';
    this.maxCount = '';
  }
}

AddStock(dat){
  this.httpService.AddData(dat)
    .subscribe(
      (data: Stock) => {this.stocks = data.arr; this.done = true; },
      error => console.log(error)
    );
  this.Update();
}

Update(){
  this.open = false;
  this.submit('Get');
  this.open = true;
}

submit(funcName: string){
  this.httpService.postData(funcName)
    .subscribe(
      (data: Stock) => {this.stocks = data.arr; this.done = true; },
      error => console.log(error)
    );
}

// Брокеры

OpenBroker(){
  if (this.openBroker === false){
    this.submitBroker('Get');
  }
  this.open = false;
  this.openSettings = false;
  this.openBroker = !this.openBroker;
}

BUpdate(){
  this.openBroker = false;
  this.submitBroker('Get');
  this.openBroker = true;
}

CloseAddBroker(){
  this.visibleB = false;
  if ( this.brokerName !== '' && this.money !== '' ){
    this.dataB = {
      name: 'Add',
      brokerName: this.brokerName,
```

```
        money: this.money
      };
      this.brokerName = '';
      this.money = '';
      this.AddBroker(this.dataB);
    }
    else {
      alert('Вы заполнели не все поля. Акции компании не были добавлены');
      this.brokerName = '';
      this.money = '';
    }
  }

  AddBroker(dat){
    this.httpService.BAddData(dat)
      .subscribe(
        (data: Broker) => {this.brokers = data.arr; this.doneB = true; },
        error => console.log(error)
      );
    this.BUpdate();
  }

  CloseB(){
    this.visibleB = false;
    this.brokerName = '';
    this.money = '';
  }

  submitBroker(funcName: string){
    this.httpService.BpostData(funcName)
      .subscribe(
        (data: Broker) => {this.brokers = data.arr; this.done = true; },
        error => console.log(error)
      );
  }

  // Настройки

  OpenSettings(){
    if(this.openSettings === false){
      this.submitSettings('Get');
    }
    this.openSettings = !this.openSettings;
    this.openBroker = false;
    this.open = false;
  }

  submitSettings(funcName: string){
    this.httpService.SpostData(funcName)
      .subscribe(
        (data: Settings) => {this.settings = data.arr; this.doneS = true; },
        error => console.log(error)
      );
  }

}
```

app.component.css

```
.container{
  max-width: 800px;
  margin: 0 auto;
}

h1{
  text-align: center;
}

.button{
  padding: 3px;
  width: 30%;
}
```

```
.addB{
  display: none;
}

.addB.okB{
  display: inline;
}

.add{
  display: none;
}

.add.ok{
  display: inline;
}

#butStock{
  position: relative;
  left: 10%;
  transform: translate(-25%, 0);
}

#butBroker{
  position: relative;
  left: 20%;
  transform: translate(-50%, 0);
}

#butSettings{
  position: relative;
  left: 30%;
  transform: translate(-75%, 0);
}
```

app.component.html

```
<div class = container>
  <h1>Биржа акций</h1>
  <hr>
  <button class="button" id="butStock" (click)="OpenStock()" bold>Акции</button>
  <button class="button" id="butBroker" (click)="OpenBroker()" >Брокеры</button>
  <button class="button" id="butSettings" (click)="OpenSettings()" bold
>Натройки</button>
  <hr>
  <div *ngIf="open">
    <button (click)="OpenAddStock()">Add</button>
    <div id="addStock" class="add" [class.ok]="visible === true">
      <p>Компания</p>
      <input id="company" value="" [(ngModel)]="company">
      <p>Количество акций</p>
      <input id="count" value="" [(ngModel)]="count" >
      <p>Цена за еденицу</p>
      <input id="price" value="" [(ngModel)]="price" >
      <p>Максимальное значение для изменения</p>
      <input id="maxCount" value="" [(ngModel)]="maxCount" >
      <p></p>
      <button (click)="CloseAddStock()">Ok</button>
      <button (click)="Close()">Закрыть</button>
    </div>
    <app-stock
      *ngFor="let iterstock of stocks"
      [stock]="iterstock"
    ></app-stock>
  </div>
  <div *ngIf="openBroker">
    <button (click)="visibleB = true">Add</button>
    <div class="addB" [class.okB]="visibleB === true">
      <p>Имя</p>
      <input [(ngModel)]="brokerName">
      <p>Счет</p>
      <input [(ngModel)]="money">
      <button (click)="CloseAddBroker()">Ok</button>
      <button (click)="CloseB()">Закрыть</button>
```

```
    </div>
    <app-broker
      *ngFor="let iterbroker of brokers"
      [broker]="iterbroker"
    ></app-broker>
  </div>
  <div *ngIf="openSettings">
    <app-setting
      *ngFor="let iterSet of settings"
      [sett]="iterSet"
     ></app-setting>
  </div>
</div>
```

http.service.ts

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';


@Injectable()
export class HttpService{

  constructor(private http: HttpClient){ }

  postData(st: string){

    const body = {name: st};
    return this.http.post('http://localhost:3000/postuser', body);
  }

  changeData(data){
    const body = {name: data.name, title: data.title, price: data.price, count:
data.count, maxCount: data.maxCount};
    return this.http.post('http://localhost:3000/postuser', body);
  }

  AddData(data){
    const body = {name: data.name, title: data.title, price: data.price, count:
data.count, maxCount: data.maxCount};
    return this.http.post('http://localhost:3000/postuser', body);
  }

  DelData(data){
    const body = {name: data.name, title: data.title, price: data.price, count:
data.count, maxCount: data.maxCount};
    return this.http.post('http://localhost:3000/postuser', body);
  }

  // Брокеры
  BchangeData(data){
    const body = {name: data.name, brokerName: data.brokerName, money:
data.money};
    return this.http.post('http://localhost:3000/broker', body);
  }

  BpostData(st: string){

    const body = {name: st};
    return this.http.post('http://localhost:3000/broker', body);
  }

  BDelData(data){
    const body = {name: data.name, brokerName: data.brokerName, money:
data.money};
    return this.http.post('http://localhost:3000/broker', body);
  }

  BAddData(data){
    const body = {name: data.name, brokerName: data.brokerName, money:
data.money};
    return this.http.post('http://localhost:3000/broker', body);
```

```
  }

  // настройки
  SpostData(st: string){

    const body = {name: st};
    return this.http.post('http://localhost:3000/set', body);
  }

  SchangeData(data){
    const body = {name: data.name, beginTimeH: data.beginTimeH, beginTimeM:
data.beginTimeM, endTimeH: data.endTimeH, endTimeM: data.endTimeM, interval :
data.interval};
    return this.http.post('http://localhost:3000/set', body);
  }
}
```

stocks.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {Stock} from '../app.component';
import {HttpService} from '../../http.service';
import {AppComponent} from '../app.component';

@Component({
  selector: 'app-stock',
  templateUrl: './stock.component.html',
  styleUrls: ['./stock.component.css'],
  providers: [HttpService]
})
export class StockComponent implements OnInit {

  constructor(private httpService: HttpService,
              private  appComponent: AppComponent){}

  @Input() stock: Stock;

  visible = false;
  done = false;
  private data: { price: string; name: string; count: string; title: string;
maxCount: string };

  OpenChange(){
    this.visible = true;
  }

  CloseChange(){
    this.visible = false;

    this.data  = {
      name: 'Change',
      title: this.stock.title,
      price: this.stock.price,
      count: this.stock.count,
      maxCount: this.stock.maxCount
    };

    this.httpService.changeData(this.data)
      .subscribe(
        (data: Stock) => {this.done = true;},
        error => console.log(error)
      );
  }

  Del(){
    this.appComponent.open = true;
    this.data  = {
      name: 'Del',
      title: this.stock.title,
      price: this.stock.price,
      count: this.stock.count,
      maxCount: this.stock.maxCount
    };
```

```
    this.httpService.DelData(this.data)
      .subscribe(
        (data: Stock) => {this.done = true; },
        error => console.log(error)
      );

    this.appComponent.Update();
  }

  ngOnInit(): void {
  }

}
```

stocks.component.css

```
.stock{
  padding: .3rem 1rem;
  border: 2px solid grey;
  margin-bottom: 1rem;
}

h3{
  text-align: center;
}

.change{
  display: none;
}
.change.ok{
  display: inline;
}
```

stocks.component.html

```
<div class="stock">
  <h3>{{ stock.title }}</h3>
<!--  <input id="title" class="change" [class.ok]="visible === true"
[(ngModel)]="stock.title">-->
  <p>Цена за 1: {{ stock.price }}</p>
  <input class="change" [class.ok]="visible === true" [(ngModel)]="stock.price">
  <p>Количество акций: {{ stock.count }}</p>
  <input class="change" [class.ok]="visible === true" [(ngModel)]="stock.count">
  <p>Закон распределения: Нормальный</p>
  <p>Максимальное значение для изменения: {{ stock.maxCount }}</p>
  <input class="change" [class.ok]="visible === true"
[(ngModel)]="stock.maxCount">
  <p></p>
  <button class="change" (click)="CloseChange()" [class.ok]="visible ===
true">Ok</button>
  <button (click)="OpenChange()">Редактировать</button>
  <p></p>
  <button (click)="Del()">Удалить</button>
</div>
```

broker.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {HttpService} from '../../http.service';
import {Broker, Stock} from '../app.component';
import {AppComponent} from '../app.component';

@Component({
  selector: 'app-broker',
  templateUrl: './broker.component.html',
  styleUrls: ['./broker.component.css'],
  providers: [HttpService]
})
export class BrokerComponent implements OnInit {

  private data: { money: string; name: string; brokerName: string };

  constructor(private httpService: HttpService, private  appComponent:
```

```
AppComponent){ }

  @Input() broker: Broker;

  visible = false;
  done = false;

  CloseChange(){
    this.visible = false;

    this.data  = {
      name: 'Change',
      brokerName: this.broker.brokerName,
      money: this.broker.money,
    };

    this.httpService.BchangeData(this.data)
      .subscribe(
        (data: Broker) => {this.done = true; },
        error => console.log(error)
      );
  }

  Del(){
    this.appComponent.openBroker = true;
    this.data  = {
      name: 'Del',
      brokerName: this.broker.brokerName,
      money: this.broker.money
    };
    this.httpService.BDelData(this.data)
      .subscribe(
        (data: Broker) => {this.done = true; },
        error => console.log(error)
      );

    this.appComponent.BUpdate();
  }

  ngOnInit(): void {
  }

}
```

broker.component.css

```
.broker{
  padding: .3rem 1rem;
  border: 2px solid grey;
  margin-bottom: 1rem;
}

.change{
  display: none;
}
.change.ok{
  display: inline;
}
```

broker.component.html

```
<div class="broker">
  <h3>{{broker.brokerName}}</h3>
  <p>Счет: {{broker.money}}</p>
  <input class="change" [class.ok]="visible === true"
[(ngModel)]="broker.money">
  <p></p>
  <button class="change" [class.ok]="visible === true"
(click)="CloseChange()">ok</button>
  <p></p>
  <button (click)="visible = true">Редактировать</button>
  <p></p>
  <button (click)="Del()">Удалить</button> </div>
```

settings.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {HttpService} from '../../http.service';
import {AppComponent} from '../app.component';
import {Settings} from '../app.component';

@Component({
  selector: 'app-setting',
  templateUrl: './setting.component.html',
  styleUrls: ['./setting.component.css']
})
export class SettingComponent implements OnInit {
  private data: { name: string; beginTimeH: any; endTimeH: any; interval:
string; endTimeM: any; beginTimeM: any };

  constructor(private httpService: HttpService, private  appComponent:
AppComponent){}

  @Input() sett: Settings;

  visible = false;
  done = false;

  ChangeSet(){
    this.visible = false;
    this.data  = {
      name: 'Change',
      beginTimeH: this.sett.beginTimeH,
      beginTimeM: this.sett.beginTimeM,
      endTimeH: this.sett.endTimeH,
      endTimeM: this.sett.endTimeM,
      interval: this.sett.interval
    };

    this.httpService.SchangeData(this.data)
      .subscribe(
        (data: Settings) => {this.done = true; },
        error => console.log(error)
      );
  }

  ngOnInit(): void {
  }

}
```

settings.component.css

```
.set{
  padding: .3rem 1rem;
  border: 2px solid grey;
  margin-bottom: 1rem;
}

.changeS{
  display: none;
}

.changeS.okS{
  display: inline;
}
```

settings.component.html

```
<div class="set">
  <h3>Время начала</h3>
    <p>Часы: {{sett.beginTimeH}}</p>
    <input class="changeS" [class.okS]="visible === true"
[(ngModel)]="sett.beginTimeH">
    <p>Минуты: {{sett.beginTimeM}}</p>
  <input class="changeS" [class.okS]="visible === true"
[(ngModel)]="sett.beginTimeM">
  <h3>Время окончания</h3>
```

```
    <p>Часы: {{sett.endTimeH}}</p>
    <input class="changeS" [class.okS]="visible === true"
[(ngModel)]="sett.endTimeH">
    <p>Минуты: {{sett.endTimeM}}</p>
  <input class="changeS" [class.okS]="visible === true"
[(ngModel)]="sett.endTimeM">
  <h3>Интервал</h3>
    <p>Минуты: {{sett.interval}}</p>
  <input class="changeS" [class.okS]="visible === true"
[(ngModel)]="sett.interval">
  <p></p>
  <button (click)="ChangeSet()" class="changeS" [class.okS]="visible ===
true">ok</button>
  <button (click)="visible = true">Редактировать</button>
</div>
```