Southwest University

Course Design

Couse Name Fundamentals of Database Systems

Semester 2019 - 2020 - 2

Grade 2018 Class 2

Student Name 陈南西

Student No. 222017321102054

Tutor Ya Li

School of Computer and Information Science

Contents

[1. Division of labor 3](#_Toc43736827)

[2. Overview 4](#_Toc43736828)

[3. Demand analysis 5](#_Toc43736829)

[4. Database design 6](#_Toc43736830)

[4.1 ER Diagram 6](#_Toc43736831)

[4.2 Database logical structure 6](#_Toc43736832)

[4.3 Database physical design 7](#_Toc43736833)

[5. Development tools and environment 10](#_Toc43736834)

[5.1 Development skill overview 10](#_Toc43736835)

[5.2 Reasons for development skill selection 10](#_Toc43736836)

[5.3 Middleware development mode 13](#_Toc43736837)

[5.4 Front end framework Blazor 14](#_Toc43736838)

[6. Main interface screenshots 16](#_Toc43736839)

[6.1 Not logged in 16](#_Toc43736840)

[6.2 Logged in as User 17](#_Toc43736841)

[6.3 Logged in as Administrator 20](#_Toc43736842)

[7. System flow diagram 25](#_Toc43736843)

[8. Source Code 34](#_Toc43736844)

[8.1 Database related code 34](#_Toc43736845)

[8.2 Page model code 38](#_Toc43736846)

[8.3 Page and logic code 39](#_Toc43736847)

[8.4 Dependency injection and middleware configuration 74](#_Toc43736848)

[9. System running analysis 77](#_Toc43736849)

[9.1 Not logged in 77](#_Toc43736850)

[9.2 Logged in as user 79](#_Toc43736851)

[9.3 Logged in as administrator 82](#_Toc43736852)

[10. Summary 89](#_Toc43736853)

[11. References 90](#_Toc43736854)

# 1. Division of labor

陈南西：Invented the whole idea, chose all the technical framework and solution, the analysis and design of the database, the design of web page, the logic function of the front page, the database related operation, etc.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Database Analysis | Database Design | Requirement analysis and function modules | Code implementation | Document writing | Interface design |
| 陈南西 | 100% | 100% | 100% | 100% | 100% | 100% |

# 2. Overview

Cloud computing is a hot topic in the Internet industry at present. This course design combines the reality of life and deeply analyzes the needs of users. A new cloud computing system is designed and developed. The system style is highly unified, the learning cost is extremely low, and users spend the least effort to do the most important things. The system has two roles of user and administrator. The two roles are the same way to enter the system, which greatly enhances the usability of the system. At the same time, administrators can dynamically modify system settings, such as adding, deleting, modifying system image templates and virtual switch templates, adding, editing, deleting users, viewing global transaction records, and managing all servers. The system has complete functions, clean and beautiful interface, strict authorization and certification logic, easy to use and high security.

# 3. Demand analysis

Just like the outbreak of the new coronavirus in 2020, we have to stay at home and take online lessons. We have a course called ”Computer network”, this course requires us to run both the client and the server at the same time. A large number of students express their sadness in our class. Some students have a slow internet speed, and download the installation image of system may cost half a day. Some students have never used a virtual machine before, It takes a long time to create a virtual machine. Some students have bad computer configuration or performance, and some students even don't have enough hard disk space, no way to do any experiments. So, because I have seen the real problems in our life and the urgent needs of everyone, I think that the system we are going to develop this time is the good news for students and which is the key to leading the reform of computer related courses in the future.

Our 25 teaching building in Southwest University have a lot of great computer resources, but now we can't go back to school because of the epidemic. Our high quality computer resources have been wasted for several months. Imagine that if our school or College of Computer and Information Science uses our cloud computing system, what does this mean? This means that our students can easily use the school's computer resources at home, to complete a variety of experiments and tasks. The computer resources originally provided for students in the school will not be idle and wasted any more.

# 4. Database design

## 4.1 ER Diagram

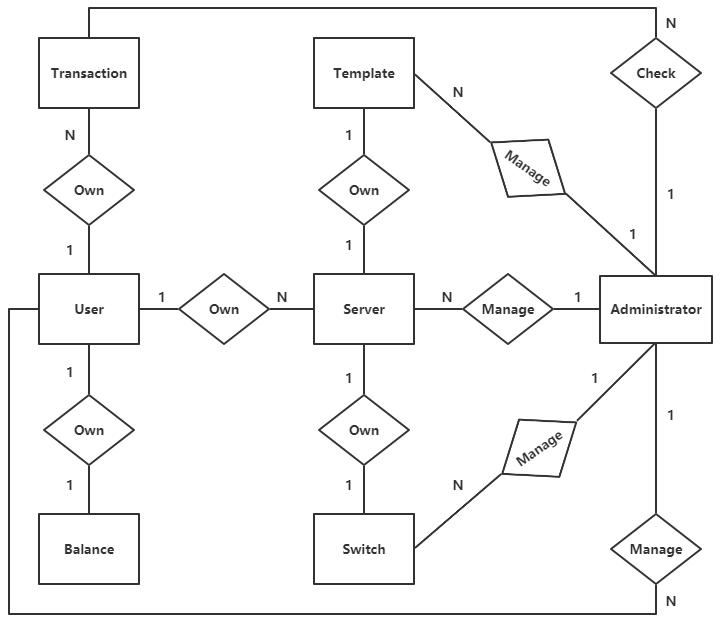


Figure ER Diagram

## 4.2 Database logical structure

Underline indicate primary key

1. AspNetRoles

(Id, Name, NormalizedName, ConcurrencyStamp)

1. AspNetUsers

(Id, UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount)

1. balance table

(username, amount)

1. server table

(serverid, username, servername, os, ip, cpu, memory, disk, expire, state, bandwidth, switchname)

1. switch table

(name, virtualswitch)

1. template table

(name, path, password)

1. transaction table

(id, username, type, amount, time, note)

## 4.3 Database physical design

Table AspNetRoles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| Id | String | False | True | Primary key |
| Name | String | True | False |  |
| NormalizedName | String | True | False |  |
| ConcurrencyStamp | String | True | False |  |
|  |  |  |  |  |

Table AspNetUsers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| Id | String | False | True | Primary Key |
| UserName | String | True | False |  |
| NormalizedUserName | String | True | False |  |
| Email | String | True | False |  |
| NormalizedEmail | String | True | False |  |
| EmailConfirmed | Bool | False | False |  |
| PasswordHash | String | True | False |  |
| SecurityStamp | String | True | False |  |
| ConcurrencyStamp | String | True | False |  |
| PhoneNumber | String | True | False |  |
| PhoneNumberConfirmed | Bool | False | False |  |
| TwoFactorEnabled | Bool | False | False |  |
| LockoutEnd | String | True | False |  |
| LockoutEnabled | Bool | False | False |  |
| AccessFailedCount | Integer | False | False |  |

Table balance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| username | String | False | True |  |
| amount | Float | False | False | balance |
|  |  |  |  |  |

Table server

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| serverid | Integer | False | True | Primary key |
| username | String | True | False |  |
| servername | String | True | False |  |
| os | String | True | False | System |
| ip | String | True | False |  |
| cpu | Integer | False | False |  |
| memory | Integer | False | False | Unit: MB |
| disk | Integer | False | False | Unit: GB |
| expire | Datetime | False | False |  |
| state | String | True | False | Running or Off |
| bandwidth | Integer | False | False | Unit: Mbps |
| switchname | String | True | False |  |
|  |  |  |  |  |

Table switch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| name | String | False | True | Primary Key |
| virtualswitch | String | True | False |  |
|  |  |  |  |  |

Table template

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| name | String | False | True | Primary Key |
| path | String | True | False | Template path |
| password | String | True | False |  |
|  |  |  |  |  |

Table transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed | Data Type | Allow Null | Key | Note |
| id | Integer | False | True | Primary Key |
| username | String | True | False |  |
| type | String | True | False | Charge/Create |
| amount | Float | False | False |  |
| time | Datetime | False | False |  |
| note | String | True | False |  |
|  |  |  |  |  |

# 5. Development tools and environment

## 5.1 Development skill overview

Development tools: Visual Studio 2019

Programming language: C#

Scripting language: PowerShell core 7

Development framework: ASP.NET Core 3

Data framework: Entity Framework core

Virtual machine engine: Hyper-V

Front end style frame: bootstrap 4

Front end interaction framework: Blazor

Server running environment: Windows system

Client running environment: Browser

## 5.2 Reasons for development skill selection

1. Virtual machine engine

The system I developed this time is "cloud computing platform server management system". For us, the server is a virtual machine with public IP address. So our first thing is to choose a suitable virtual machine engine. Because most of the systems we usually use are Windows systems, and my own computer is also installed with windows, so if I deliberately use the virtual machine engine under the Linux system, it is a bit inappropriate, not convenient for our development, testing and experience. Considering the actual situation, I chose Hyper-V, the virtual machine engine that comes with Windows system. In fact, Hyper-V is free. We can download the independent version of Hyper-V from Microsoft's website without license restrictions. Therefore, there is no commercial and license related dispute when we use Hyper-V.

1. Programming language

Because I develop independently, I pay more attention to efficiency in technical options, so I choose C# language, the syntax of C# language is advanced, and C# is the official programming language recommended by Microsoft. It works very well with products under the Microsoft system. Therefore, C# is suitable for our development this time.

1. Database access

We don't care what database program we choose this time, because we use the database access framework recommended by Microsoft called “Entityframework core”, or EF core for short.

EF core is an open source framework under the .Net system, which implements the abstraction of database operation. Because all databases have the function of adding, deleting, querying and modifying, EF core has set up a standard, under which the syntax and command of database operation are the same. If we want to use mysql, we will use the corresponding EF of MySQL Core, if we want to use Oracle, we will use the EF core corresponding to Oracle. Because the usage of EF core is uniform, our code can switch between different databases without any changes, which is very flexible.

1. Web development

In order to learn the latest development ideas, so when I choose the website development technology scheme, I prefer to pursue cutting-edge and new technology. The website development mode I chose this time is Microsoft's latest .Net core Blazor framework. Unlike the traditional framework, this framework does not simply help developers encapsulate some functions and codes, but innovates their development ideas. For example, to develop a large-scale dynamic website, we must use JavaScript, but under the Blazor framework, we can use C# language to develop the front-end without writing any JavaScript code. I wrote this sentence in a plain way. In fact, this is the first framework of this type, which is very innovative. In order to develop websites, former developers had to learn a JavaScript language separately, which undoubtedly increased the learning cost of developers. Now with the development of web-assembly technology, more and more high-level languages can replace JavaScript to develop the front-end of the website to achieve the effect of dynamic website. Therefore, in order to experience the latest web development framework and ideas, I chose the Blazor framework.

1. Scripting language

Scripting language is the language for users to deal with the system, while the mainstream programming language generally deals with business logic, so it is inconvenient and inflexible to use programming language to deal with the system. In order to solve this problem, we need to cooperate with scripting language to implement the operation. Because we are developing under the windows platform, I choose the PowerShell scripting language directly here. The development direction of PowerShell today is cross platform, and the idea of PowerShell is very different from that of Linux. The shell under Linux is text-oriented. If we want to get a specific value, we have to use many strange techniques to deal with string text. With PowerShell, we can easily get the value we want with the object-oriented programming idea, instead of focusing on the process of output results. In addition, PowerShell's command naming is standard and easy to identify and read, unlike bash scripts on Linux, which is hard to understand.

1. Front end style framework

I choose Bootstrap because Bootstrap is used by default in the ASP.NET CORE. Secondly, Bootstrap is really easy to use and learn. Official documents are friendly and provide many code with many effect or appearance. Suppose we want a good-looking button, we will go to the document to select the button example, as shown in the following figure:

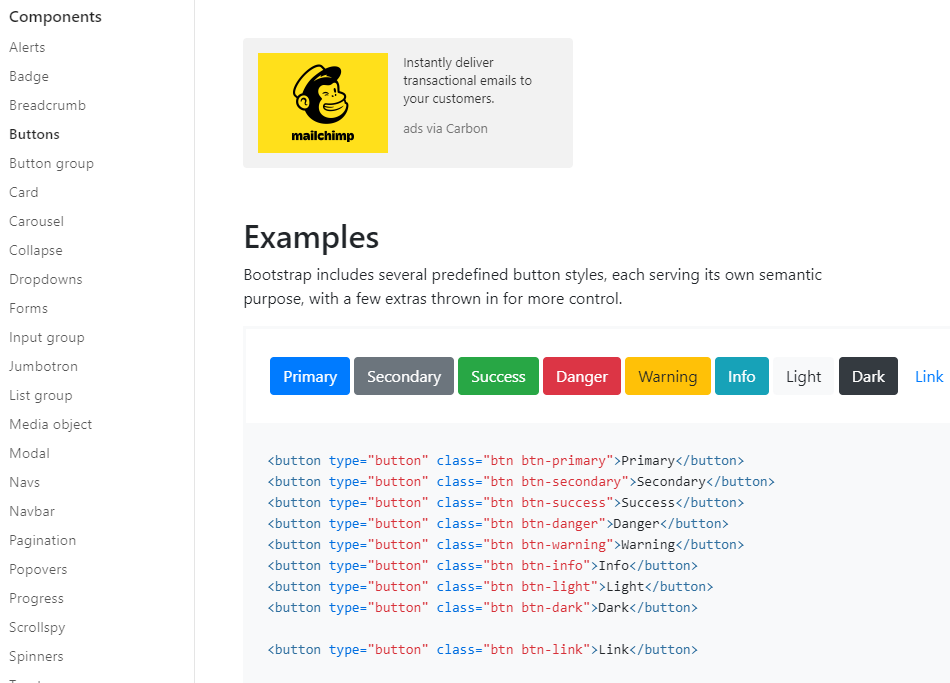


Figure Bootstrap - Button Example

1. Development platform

I chose .Net Core as our development platform because. Net Core is the main focus of Microsoft. There are many old and traditional frameworks developed by Microsoft. Different frameworks have been developed for different tasks. Gradually, Microsoft realizes that the cost of maintaining these frameworks is getting higher and higher, which is more and more troublesome. Therefore, Microsoft provides huge human and material resources to solve this problem, so they choose to reconstruct .Net Framework, which implements advanced and modular .Net core platform or framework. According to Microsoft's plan, the .Net core 5 version will be called .Net 5 for short, because in the future .Net framework will not be changed greatly. Although .Net 5 and later versions do not add “core”, they are actually .Net Core by default, which means that .Net Core framework will unify the major development platforms and technology stacks later.

In short, in the future, Microsoft will provide complete development tools, complete solutions, cross platform development with a unified framework .Net in various fields. It means that .Net developers can develop websites, mobile apps, desktop programs, games, Internet of things applications, artificial intelligence and so on.

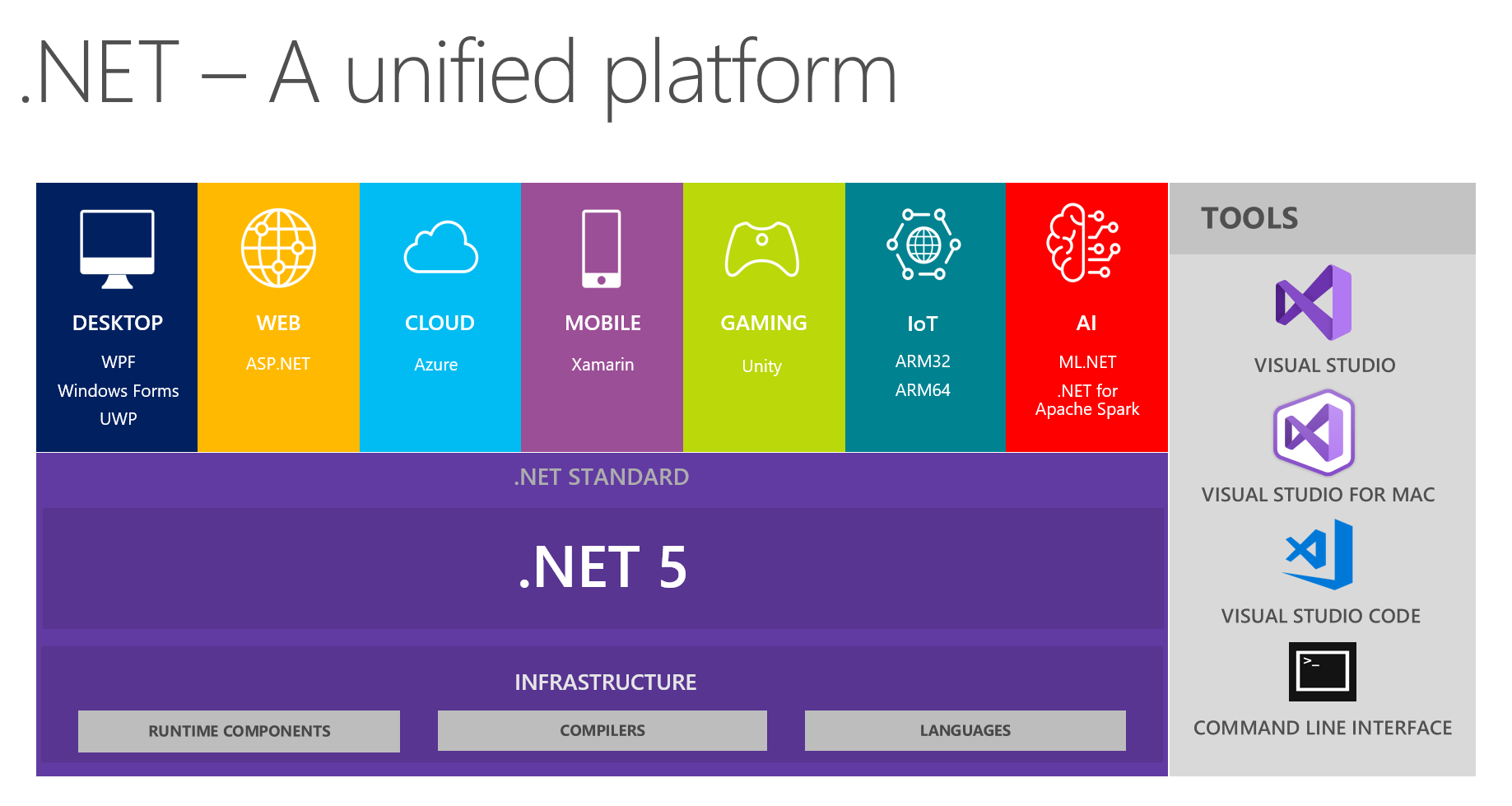


Figure Unified .Net platform

## 5.3 Middleware development mode

Middleware is software that's assembled into an app pipeline to process requests and responses. Each component:

* Chooses whether to pass the request to the next component in the pipeline.
* Can perform task before and after the next component in the pipeline.

Request delegates are used to build the request pipeline. The request delegates process each HTTP request.

Each middleware component in the request pipeline is responsible for invoking the next component in the pipeline or short-circuiting the pipeline. When a middleware short-circuits, it's called a *terminal middleware* because it prevents further middleware from handling the request.

The ASP.NET Core request pipeline consists of a sequence of request delegates, called one after the other. The following diagram demonstrates the concept. The thread of execution follows the black arrows.



Figure Middleware pipeline

We can decide and rearrange the order of middleware

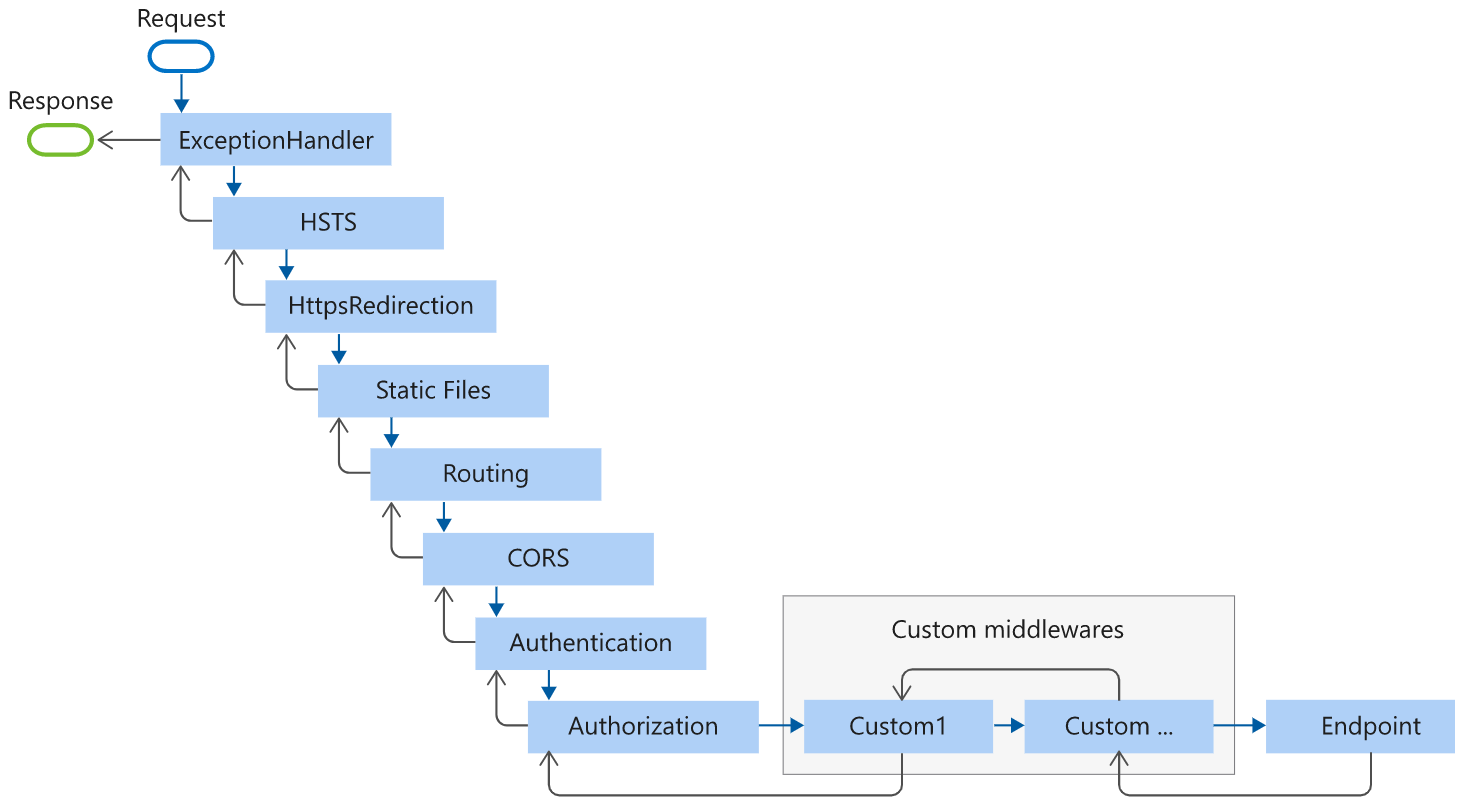
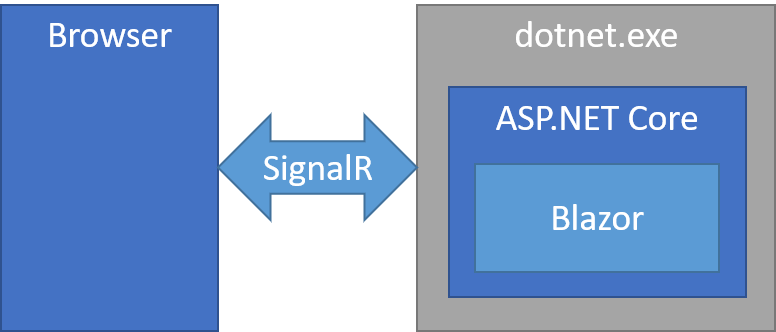


Figure Middleware order

## 5.4 Front end framework Blazor

Using the Blazor server hosting model, ASP.NET Core application can execute on the server. UI updates, event handling, and JavaScript calls are handled through a SignalR connection.



A large number of generic lists and collections are used in the code of interaction between the system and the database, which inherits the interface of IEnumerable, so that our generic lists and collections can be read by foreach loop. Then our page uses foreach loop to generate entity objects to the page one by one, and generate corresponding operation buttons. After clicking the button, we will call the delegation and events in .Net, and send the entity object in the page as a parameter to the target method delegation for function call through lamda expression to execute corresponding operation.

# 6. Main interface screenshots

## 6.1 Not logged in

Default Homepage:



Figure Default Homepage

Register Page:

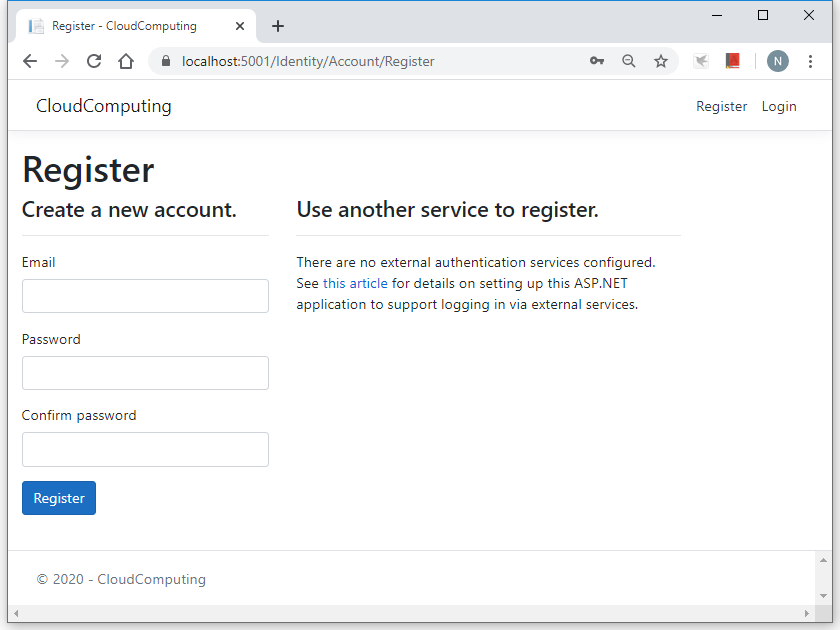


Figure Register page

Login Page:

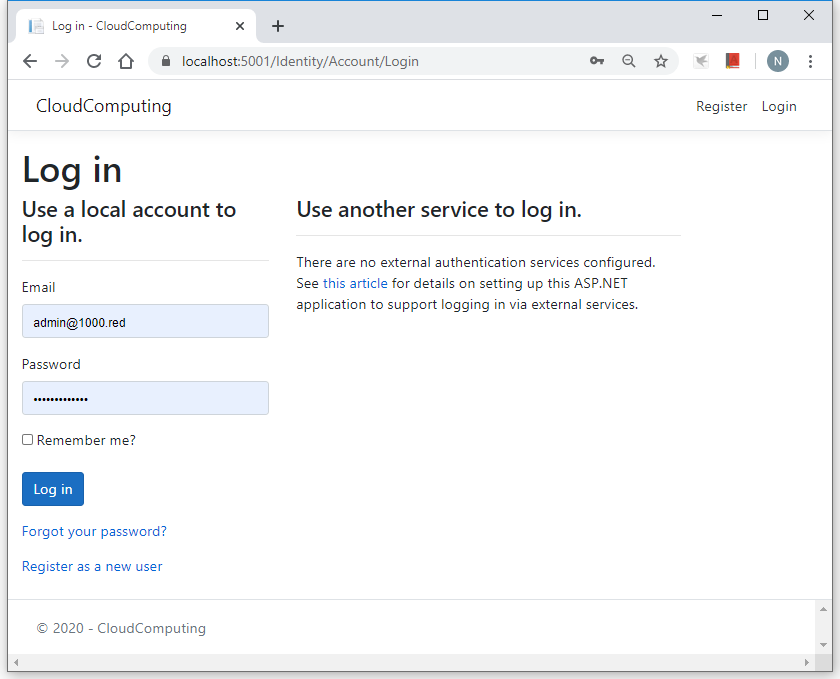


Figure Login Page

## 6.2 Logged in as User

Top Up Page:

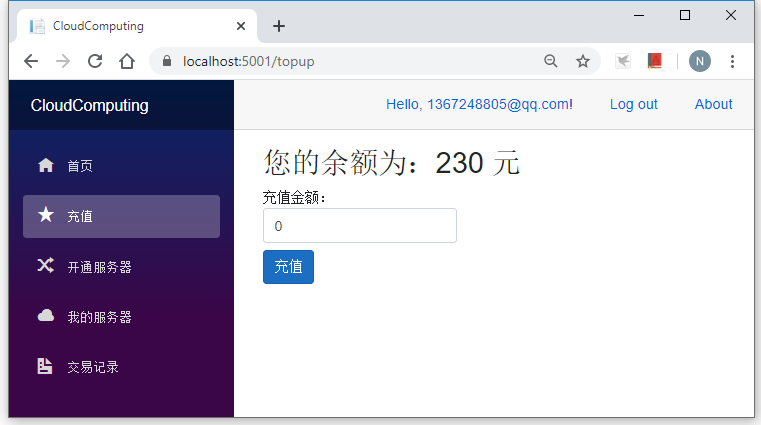


Figure Top Up Page

Create Server Page:

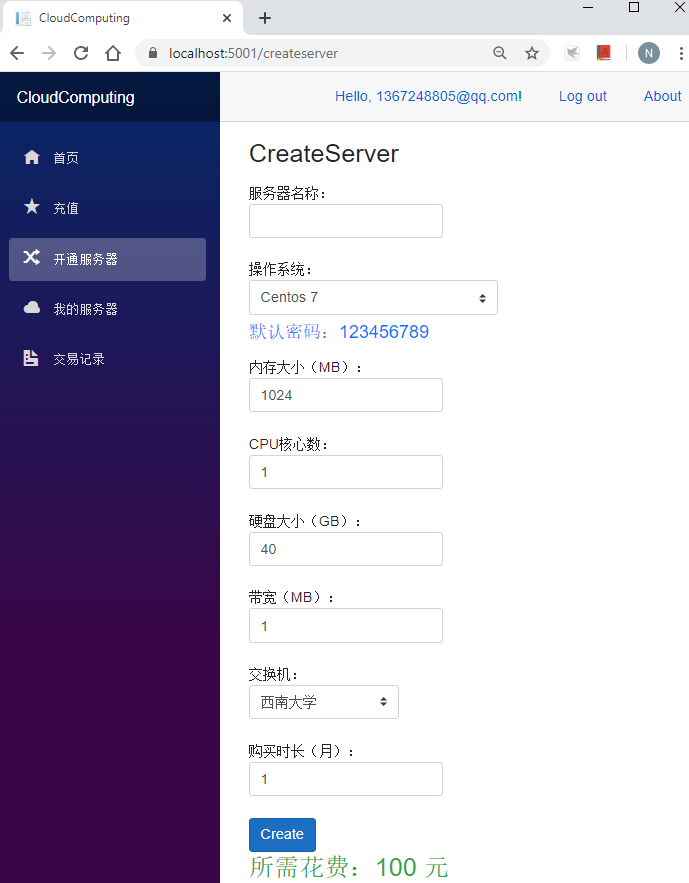


Figure Create Server Page

Server List Page:

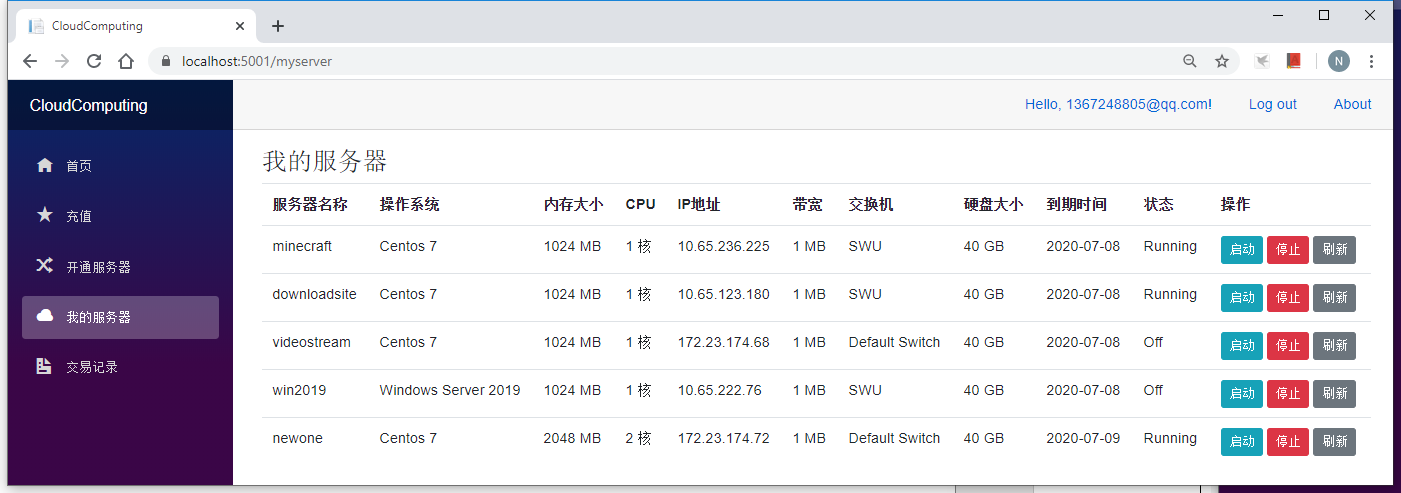


Figure Server List Page

Transaction Record Page:

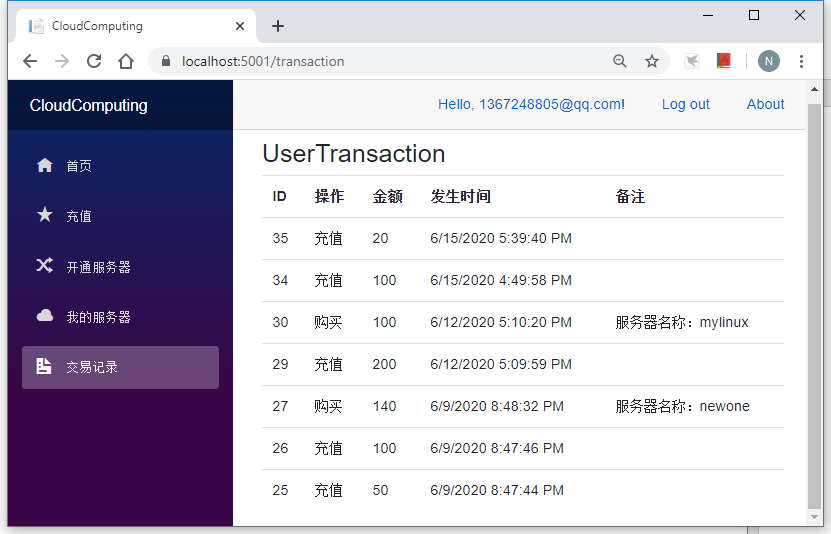


Figure Transaction Record Page

## 6.3 Logged in as Administrator

Administrator Homepage:

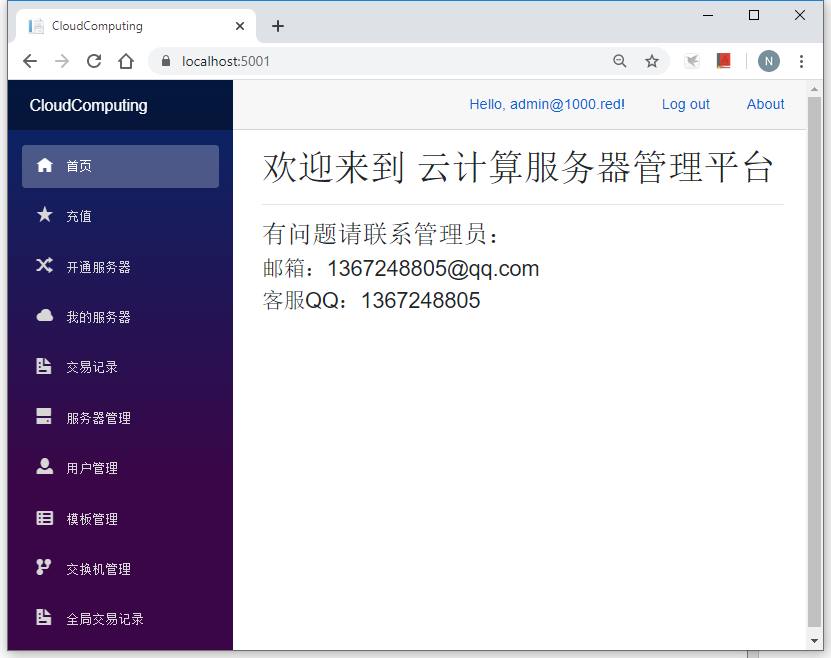


Figure Administrator Homepage

Server Management Page:

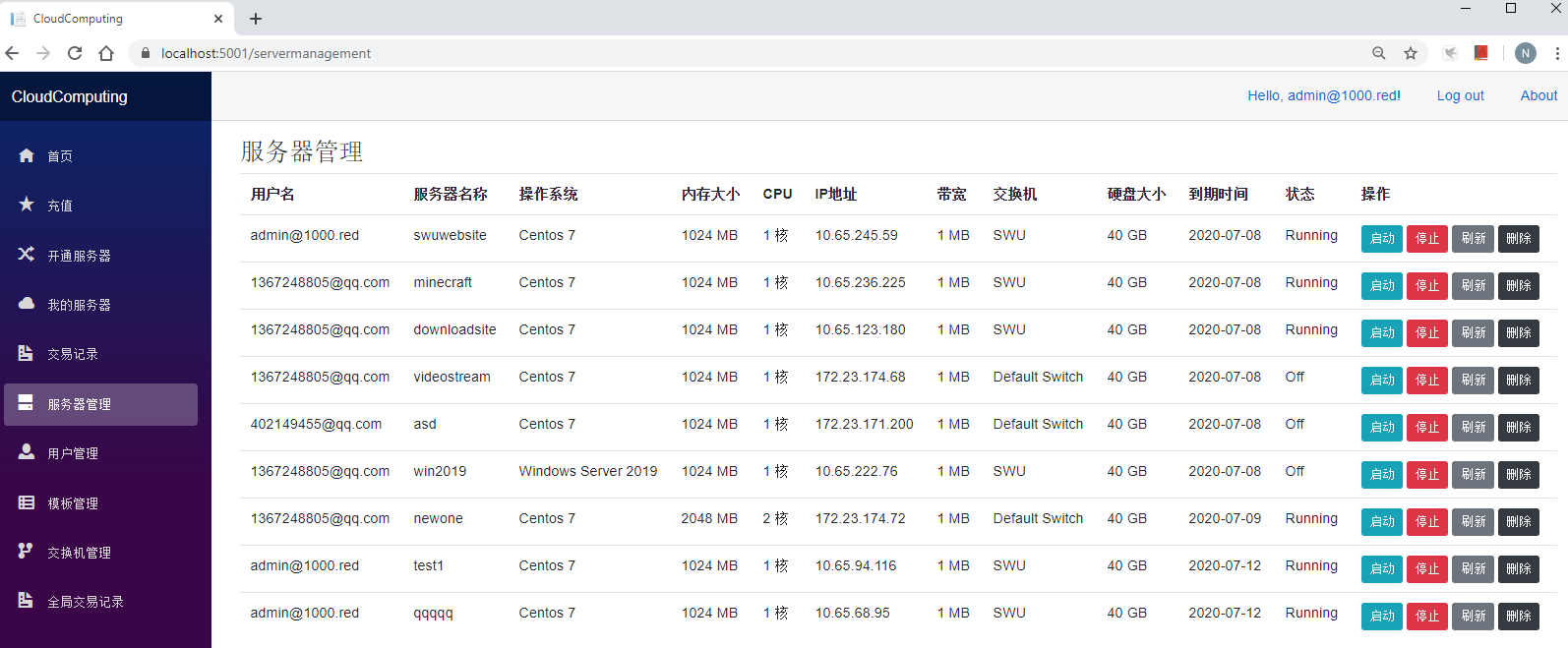


Figure Server Management Page

User Management Page:

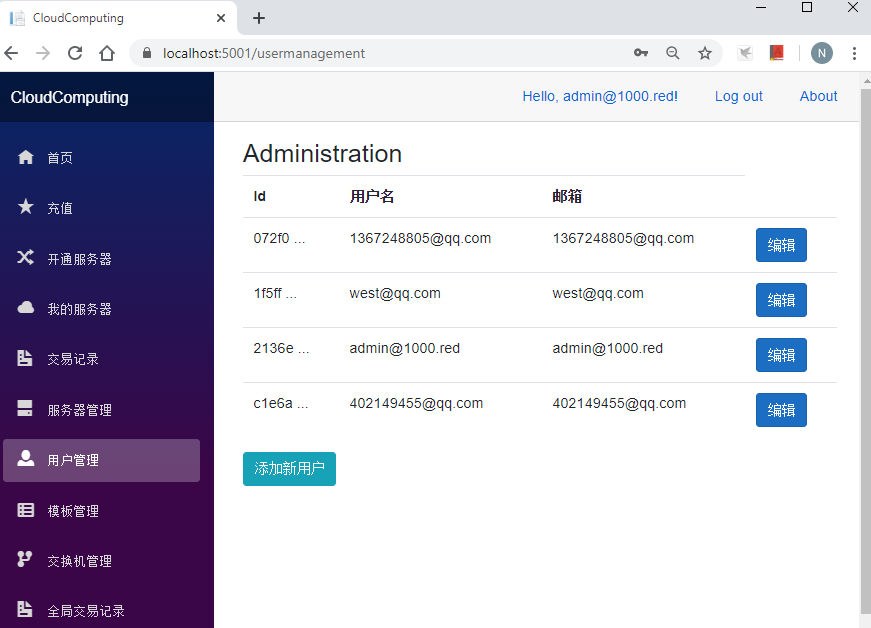


Figure User Management

Operating System Template Management Page:

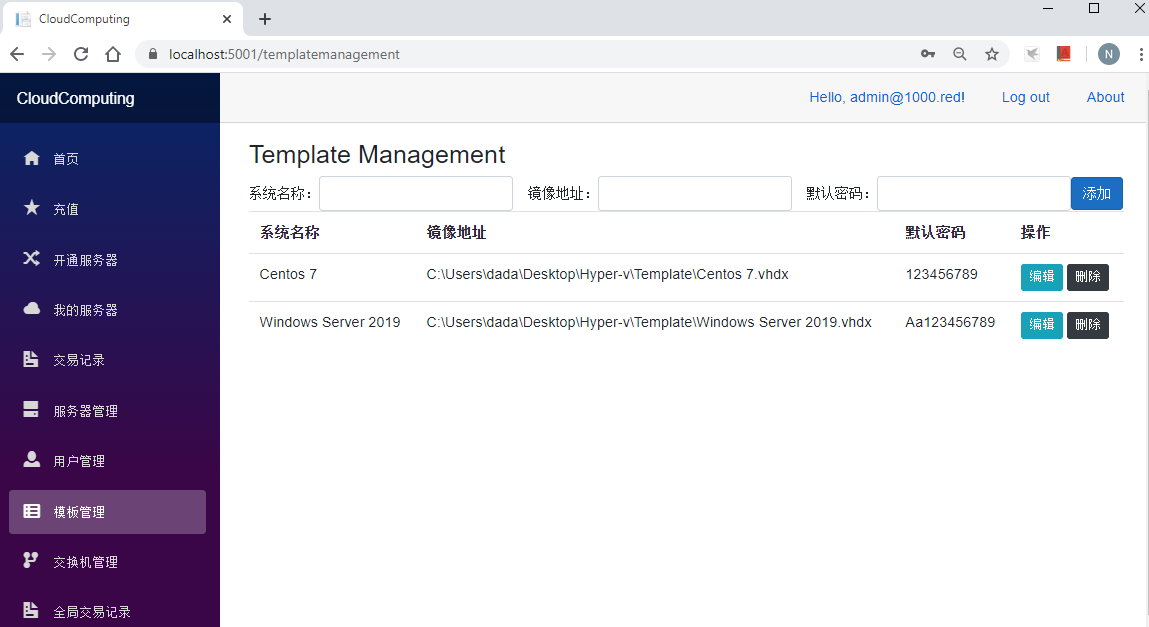


Figure OS Template Management Page

Switch Management Page:

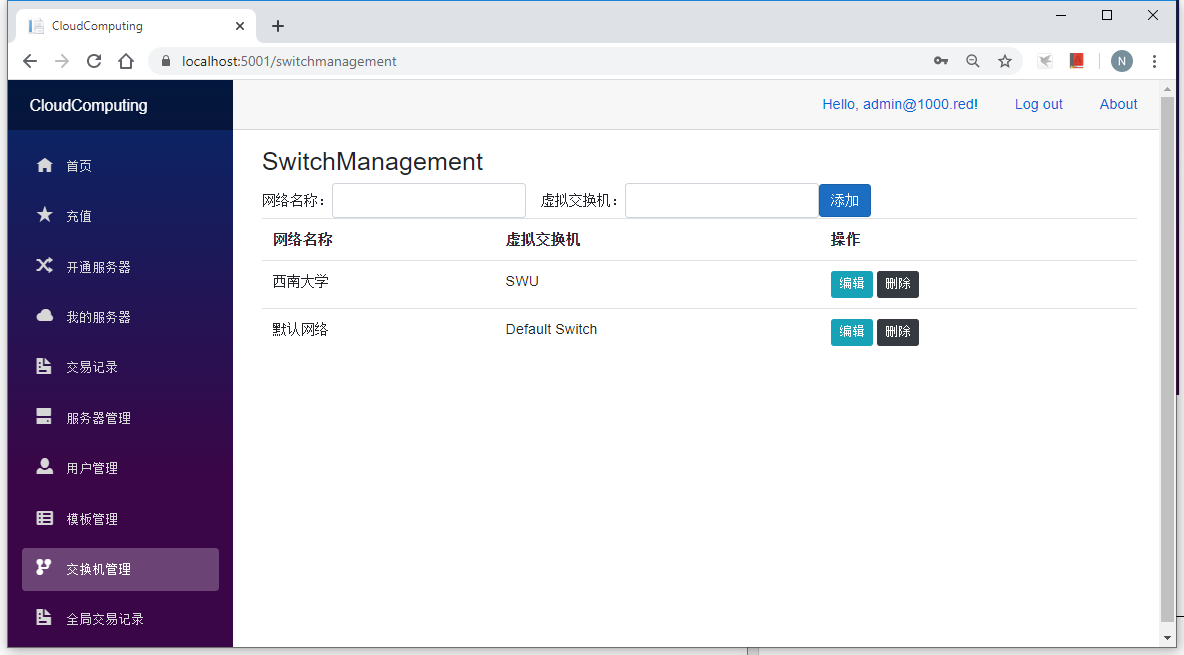


Figure Switch Management Page

All Transaction Record Page:

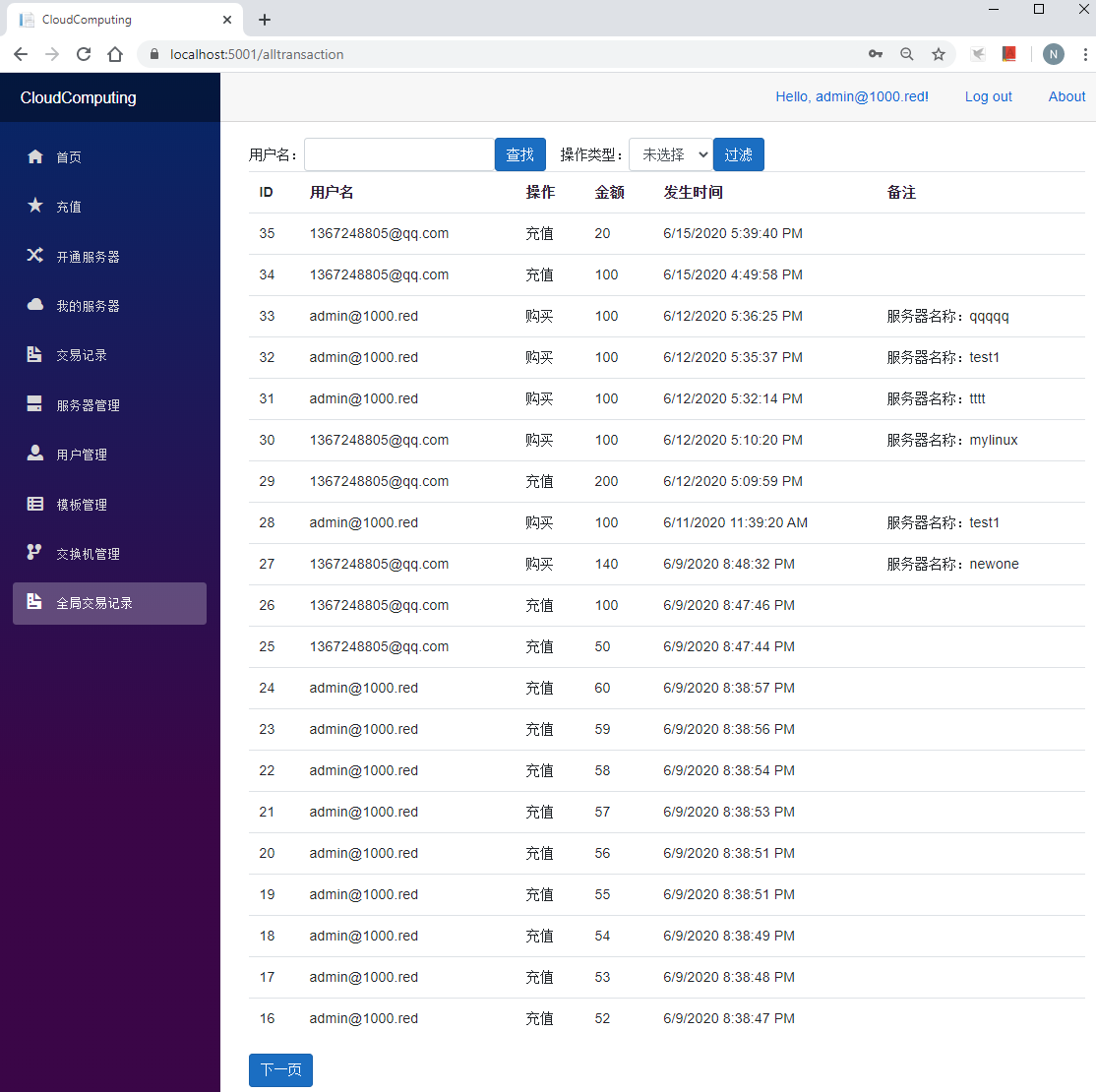


Figure All Transaction Record Page

# 7. System flow diagram

Top Up:

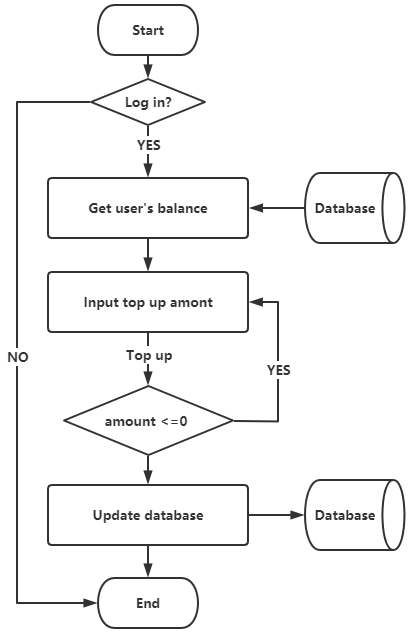


Figure Top Up flow diagram

Create Server:

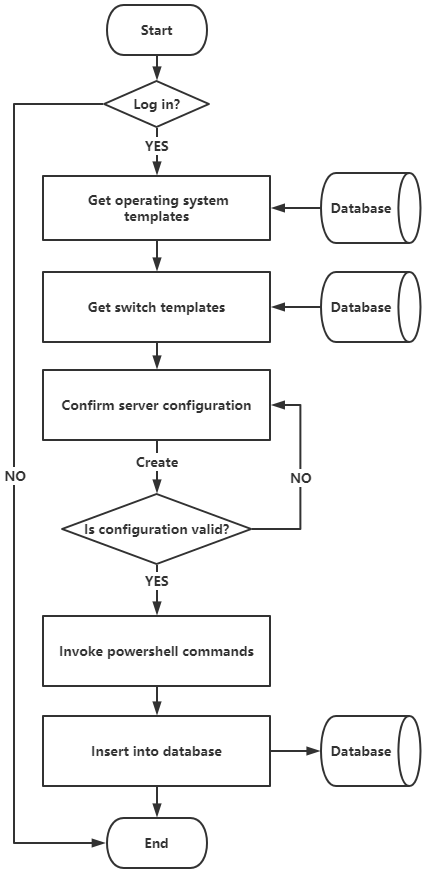


Figure Create Server flow diagram

My Server:

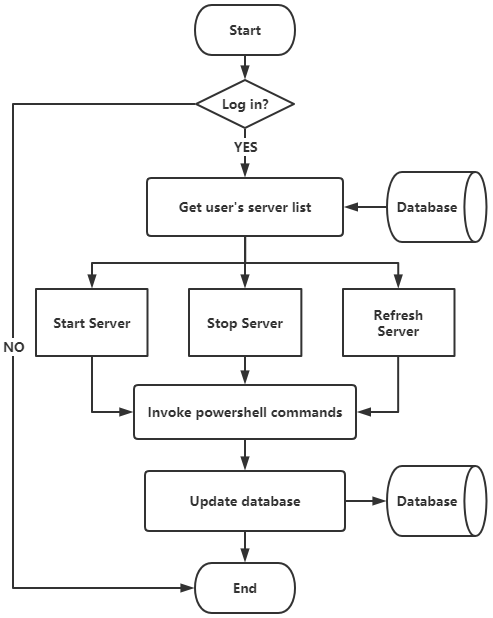


Figure My Server flow diagram

Transaction Record:

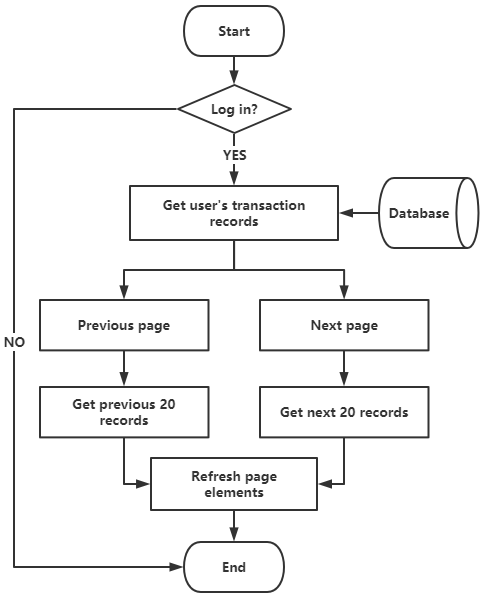


Figure Transaction Record flow diagram

Server Management:

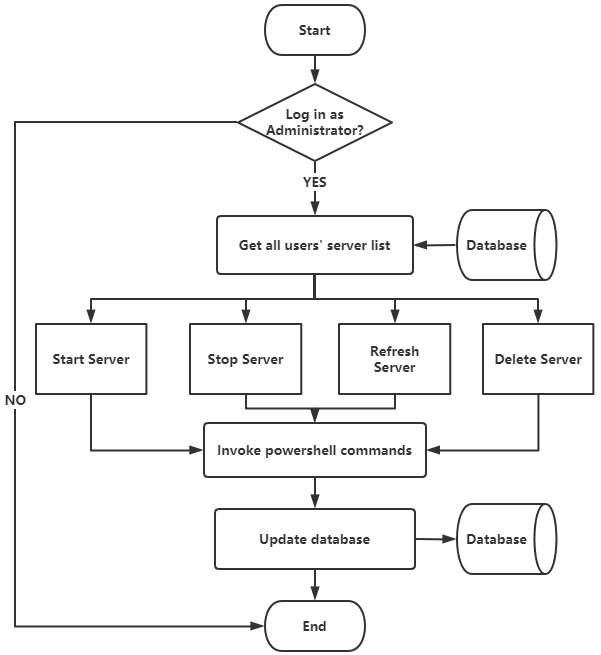


Figure Server Management flow diagram

User Management:

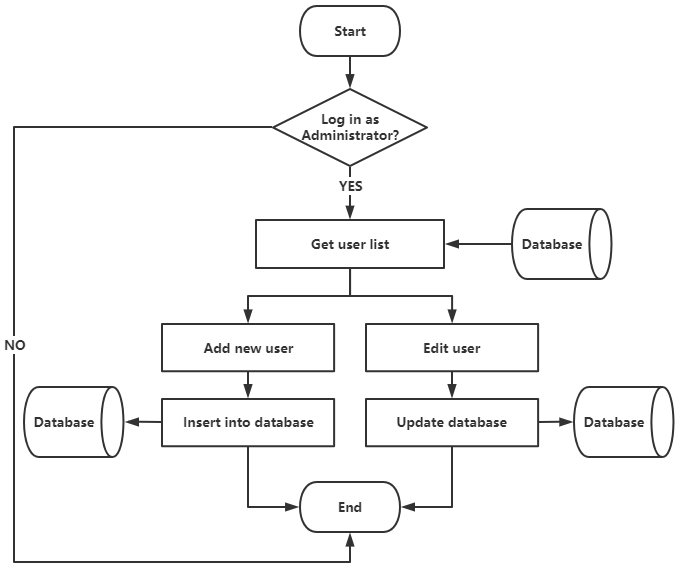


Figure User Management flow diagram

OS Template Management:

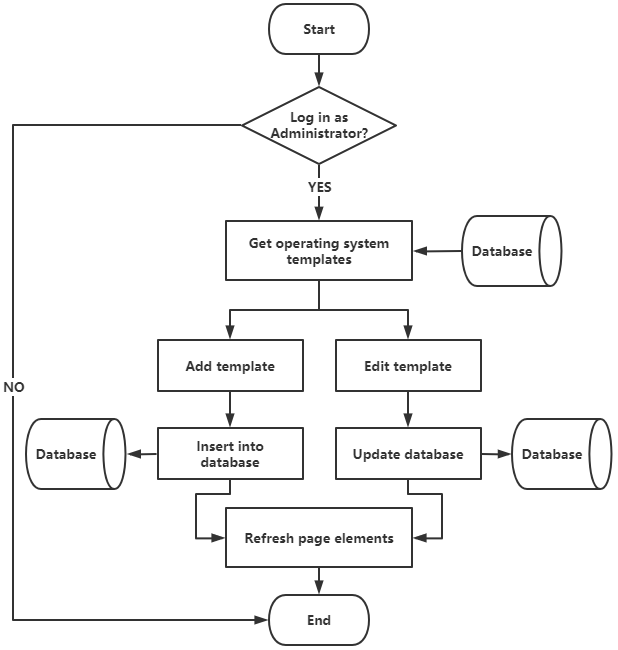


Figure OS Template Management flow diagram

Switch Management:

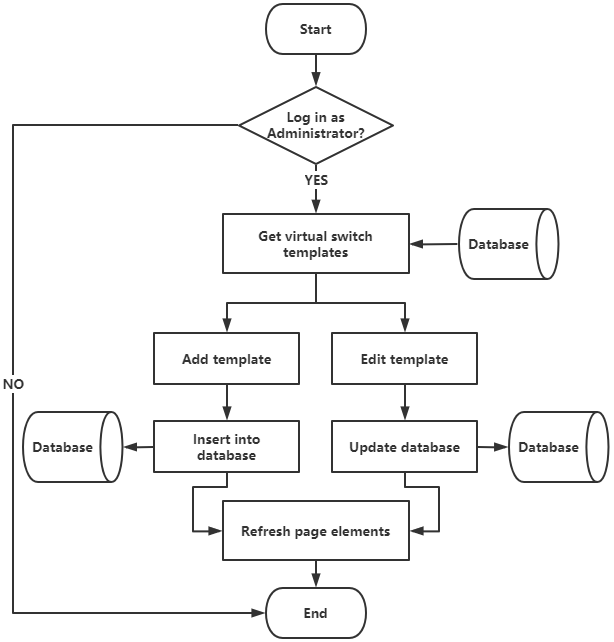


Figure Switch Management flow diagram

All Transaction Record:

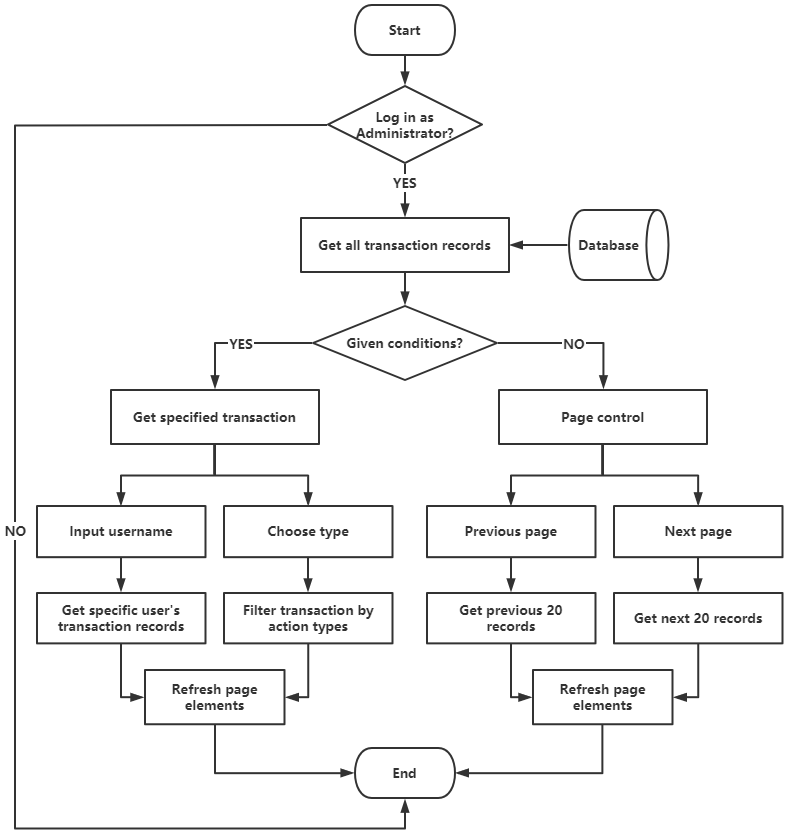


Figure All Transaction Record flow diagram

# 8. Source Code

## 8.1 Database related code

**Database access context class: ApplicationDbContext.cs**

using System;

using System.Collections.Generic;

using System.Text;

using CloudComputing.Model;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace CloudComputing.Data

{

public class ApplicationDbContext : IdentityDbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Balance> balances { get; set; }

public DbSet<Server> servers { get; set; }

public DbSet<Transaction> transactions { get; set; }

public DbSet<Template> templates { get; set; }

public DbSet<Switch> switches { get; set; }

}

}

**Balance database table class: Balance.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

[Table("balance")]

public class Balance

{

[Column("username")][Key][Required]

public string username { get; set; }

[Column("amount")][Required]

public double amount { get; set; }

}

}

**Server database table class: Server.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

[Table("server")]

public class Server

{

[Column("username")]

public string username { get; set; }

[Column("servername")]

public string servername { get; set; }

[Column("serverid")]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int serverid { get; set; }

[Column("os")]

public string os { get; set; }

[Column("ip")]

public string ip { get; set; }

[Column("cpu")]

public int cpu { get; set; }

[Column("memory")]

public float memory { get; set; }

[Column("bandwidth")]

public int bandwidth { get; set; }

[Column("switchname")]

public string switchname { get; set; }

[Column("state")]

public string state { get; set; }

[Column("disk")]

public int disk { get; set; }

[Column("expire")]

public DateTime expire { get; set; }

}

}

**Switch database table class: Switch.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

[Table("switch")]

public class Switch

{

[Key]

[Column("name")]

public string name { get; set; }

[Column("virtualswitch")]

public string virtualswitch { get; set; }

}

}

**Template database table class: Template.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

[Table("template")]

public class Template

{

[Key]

[Column("name")]

public string name { get; set; }

[Column("path")]

public string path { get; set; }

[Column("password")]

public string password { get; set; }

}

}

**Transaction database table class: Transaction.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

[Table("transaction")]

public class Transaction

{

[Column("id")]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int id { get; set; }

[Column("username")]

public string username { get; set; }

[Column("type")]

public string type { get; set; } //充值、购买

[Column("amount")]

public double amount { get; set; }

[Column("time")]

public DateTime time { get; set; }

[Column("note")]

public string note { get; set; }

}

}

## 8.2 Page model code

**Create server verification model: CreateServerModel.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

namespace CloudComputing.Model

{

public class CreateServerModel

{

[Required]

[StringLength(20, ErrorMessage = "Server name is too long.")]

public string servername { get; set; }

[Required]

public string os { get; set; }

[Required]

public int memory { get; set; }

[Required]

public int bandwidth { get; set; }

[Required]

public string switchname { get; set; }

[Required]

public int cpu { get; set; }

[Required]

public int disk { get; set; }

}

}

## 8.3 Page and logic code

**Layout component: MainLayout.razor**

@inherits LayoutComponentBase

<div class="sidebar">

<**NavMenu** />

</div>

<div class="main">

<div class="top-row px-4 auth">

<**LoginDisplay** />

<a href="https://docs.microsoft.com/aspnet/" target="\_blank">About</a>

</div>

<div class="content px-4">

@Body

</div>

</div>

**Login component: LoginDisplay.razor**

<**AuthorizeView**>

<**Authorized**>

<a href="Identity/Account/Manage">Hello, @context.User.Identity.Name!</a>

<form method="post" action="Identity/Account/LogOut">

<button type="submit" class="nav-link btn btn-link">Log out</button>

</form>

</**Authorized**>

<**NotAuthorized**>

<a href="Identity/Account/Register">Register</a>

<a href="Identity/Account/Login">Log in</a>

</**NotAuthorized**>

</**AuthorizeView**>

**Navigation bar: NavMenu.razor**

<div class="top-row pl-4 navbar navbar-dark">

<a class="navbar-brand" href="">CloudComputing</a>

<button class="navbar-toggler" **@onclick**="ToggleNavMenu">

<span class="navbar-toggler-icon"></span>

</button>

</div>

<div class="@NavMenuCssClass" **@onclick**="ToggleNavMenu">

<ul class="nav flex-column">

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="" **Match**="NavLinkMatch.All">

<span class="oi oi-home" aria-hidden="true"></span> 首页

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="topup">

<span class="oi oi-star" aria-hidden="true"></span> 充值

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="createserver">

<span class="oi oi-random" aria-hidden="true"></span> 开通服务器

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="myserver">

<span class="oi oi-cloud" aria-hidden="true"></span> 我的服务器

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="transaction">

<span class="oi oi-document" aria-hidden="true"></span> 交易记录

</**NavLink**>

</li>

<**AuthorizeView** **Roles**="Administrators">

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="servermanagement">

<span class="oi oi-hard-drive" aria-hidden="true"></span> 服务器管理

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="usermanagement">

<span class="oi oi-person" aria-hidden="true"></span> 用户管理

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="templatemanagement">

<span class="oi oi-spreadsheet" aria-hidden="true"></span> 模板管理

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="switchmanagement">

<span class="oi oi-fork" aria-hidden="true"></span> 交换机管理

</**NavLink**>

</li>

<li class="nav-item px-3">

<**NavLink** class="nav-link" href="alltransaction">

<span class="oi oi-document" aria-hidden="true"></span> 全局交易记录

</**NavLink**>

</li>

</**AuthorizeView**>

</ul>

</div>

@code {

private bool collapseNavMenu = true;

private string NavMenuCssClass => collapseNavMenu ? "collapse" : null;

private void ToggleNavMenu()

{

collapseNavMenu = !collapseNavMenu;

}

}

**Top up page: TopUp.razor**

@page "/topup"

@inject CloudComputing.Data.ApplicationDbContext db

@using Model

@inject AuthenticationStateProvider AuthenticationStateProvider

<**AuthorizeView**>

<**Authorized**>

<h2>您的余额为：@amount 元</h2>

<label>

充值金额：

<input class="form-control" type="text" **@bind-value**="\_balance" placeholder="请输入金额" />

</label>

<br />

<button class="btn btn-primary" **@onclick**="()=>AddBalance(context.User.Identity.Name)">充值</button>

<p class="text-danger">@danger</p>

<p class="text-success">@success</p>

</**Authorized**>

<**NotAuthorized**>

<h2>请先登录</h2>

</**NotAuthorized**>

</**AuthorizeView**>

@code {

//警告信息

private string danger = "";

//成功充值信息

private string success = "";

//充值金额

private int \_balance { get; set; }

//用户当前余额

private double amount = 0;

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

//充值余额

private void AddBalance(string username)

{

if (db.balances.Find(username) == null)

{

db.balances.Add(new Balance() { username = username, amount = 0 });

}

if (\_balance <= 0)

{

danger = "充值金额不合法，请修改后再试！";

success = "";

return;

}

db.balances.Find(username).amount += \_balance;

db.SaveChanges();

amount = db.balances.Find(username).amount;

db.transactions.Add(new Model.Transaction()

{

username = username,

type = "充值",

amount = \_balance,

time = DateTime.Now,

note = ""

});

db.SaveChanges();

success = $"成功充值{\_balance}元";

\_balance = 0;

danger = "";

}

//初始化

protected override async Task OnInitializedAsync()

{

var CurrentUser = (await authenticationStateTask).User;

//如果当前没有登录用户，不作处理

if (CurrentUser.Identity.Name == null)

{

return;

}

//如果当前用户没有记录，则添加

if (db.balances.Find(CurrentUser.Identity.Name) == null)

{

db.balances.Add(new Balance() { username = CurrentUser.Identity.Name, amount = 0 });

db.SaveChanges();

}

//读取用户余额

amount = db.balances.Find(CurrentUser.Identity.Name).amount;

}

}

**Create server page: CreateServer.razor**

@page "/createserver"

@inject NavigationManager nav

@using System.Management.Automation

@using System.Management.Automation.Runspaces

@using System.Collections.ObjectModel

@using Model

@inject CloudComputing.Data.ApplicationDbContext db

@inject AuthenticationStateProvider AuthenticationStateProvider

@attribute [Authorize]

<h3>CreateServer</h3>

<**EditForm** **Model**="@serverinfo" **OnValidSubmit**="create">

<**DataAnnotationsValidator** />

<**ValidationSummary** />

<div class="form-group">

<div class="row"><label class="text-danger">@error</label></div>

<label>

服务器名称：<input class="form-control" type="text" **@bind-value**="serverinfo.servername" />

</label>

</div>

<div class="form-group">

<label>

操作系统：<select class="custom-select" **@bind**="serverinfo.os">

@foreach (var template in templates)

{

<option value="@template.name">@template.name</option>

}

</select>

</label><h5 class="text-primary">默认密码：@password</h5>

</div>

<div class="form-group">

<label>

内存大小（MB）：<input class="form-control" type="text" **@bind-value**="serverinfo.memory" />

</label>

</div>

<div class="form-group">

<label>

CPU核心数：<input class="form-control" type="text" **@bind-value**="serverinfo.cpu" />

</label>

</div>

<div class="form-group">

<label>

硬盘大小（GB）：<input class="form-control" type="number" **@bind-value**="serverinfo.disk" />

</label>

</div>

<div class="form-group">

<label>

带宽（MB）：<input class="form-control" type="number" **@bind-value**="serverinfo.bandwidth" />

</label>

</div>

<div class="form-group">

<label>

交换机：<select class="custom-select" **@bind**="serverinfo.switchname">

@foreach (var s in switches)

{

<option value="@s.virtualswitch">@s.name</option>

}

</select>

</label>

</div>

<div class="form-group">

<label>

购买时长（月）：<input class="form-control" type="number" **@bind-value**="months" />

</label>

</div>

<button class="btn btn-primary" disabled="@isDisabled" **@onclick**="BeforeCreate" type="submit">Create</button>

</**EditForm**>

<h3 class="text-success">所需花费：@cost 元</h3>

<lable class="text-danger">@message</lable>

@code {

//通过lamda表达式计算配置所需花费

private double cost => (serverinfo.memory / 1024 \* 20 //内存 1G/20元

+ serverinfo.cpu \* 20 //CPU 1核/20元

+ serverinfo.disk \* 1 //硬盘 1G/1元

+ serverinfo.bandwidth \* 20) //带宽 1MB/20元

\* months; //配置价格\*月数

//系统镜像模板列表

private List<Template> templates;

//交换机配置列表

private List<Switch> switches;

//通过lamda表达式获取对应系统镜像的默认密码

private string password => db.templates.Where(t => t.name == serverinfo.os).First().password;

//创建按钮是否被禁用

private bool isDisabled = false;

//提示信息

private string message = "";

//默认开通月数为1

private int months = 1;

//报错信息

private string error;

//创建服务器所需的信息集合

private CreateServerModel serverinfo = new CreateServerModel()

{

cpu = 1,

disk = 40,

memory = 1024,

os = "",

bandwidth = 1,

switchname = ""

};

//初始化

protected override async Task OnInitializedAsync()

{

templates = db.templates.ToList();

switches = db.switches.ToList();

if (templates.Count > 0)

{

serverinfo.os = templates.First().name;

}

if (switches.Count > 0)

{

serverinfo.switchname = switches.First().virtualswitch;

}

}

//创建服务器之前更新界面UI

private void BeforeCreate()

{

isDisabled = true;

message = "正在开通服务器，请等待...";

}

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

//系统镜像模板文件所在路径

string scrVHDpath;

//新建虚拟机的目标路径

string desVHDpath;

//保存新建虚拟机的根目录

string path = @"C:\Users\dada\Desktop\Hyper-v\VM";

//创建服务器

async Task create()

{

scrVHDpath = db.templates.Where(t => t.name == serverinfo.os).First().path;

desVHDpath = path + @"\" + serverinfo.servername + @"\" + serverinfo.servername + ".vhdx";

var s = db.servers.Where(s => s.servername == serverinfo.servername);

//能查找到意味着已经存在

if (s.Count() != 0)

{

error = "已经存在该服务器名称，请修改后重试";

message = "";

isDisabled = false;

return;

}

error = "";

var CurrentUser = (await authenticationStateTask).User;

var b = db.balances.Find(CurrentUser.Identity.Name);

if (b.amount < cost) //余额不够

{

message = "您的余额不足，请先充值";

isDisabled = false;

return;

}

b.amount -= cost;

db.SaveChanges();

PowerShell ps1 = PowerShell.Create();

//根据服务器名称创建对应目录

ps1.AddScript(@$"New-Item -Type Directory '{path + @"\" + serverinfo.servername}'");

ps1.AddScript(@$"Copy-Item -Path '{scrVHDpath}' -Destination '{desVHDpath}'");

ps1.Invoke();

//创建虚拟机并且对其进行设置

PowerShell ps2 = PowerShell.Create();

ps2.AddCommand("New-VM")

.AddParameter("Name", serverinfo.servername)

.AddParameter("VHDPath", desVHDpath)

.AddParameter("Path", path + @"\" + serverinfo.servername)

.AddParameter("SwitchName", serverinfo.switchname);

ps2.AddScript($"Set-VMMemory '{serverinfo.servername}' -DynamicMemoryEnabled $true -MaximumBytes {serverinfo.memory}GB");

ps2.AddScript($"Resize-VHD -Path {desVHDpath} -SizeBytes {serverinfo.disk}GB");

ps2.AddScript($"Set-VMProcessor '{serverinfo.servername}' -Count {serverinfo.cpu}");

ps2.AddScript($"Set-VMNetworkAdapter -VMName '{serverinfo.servername}' -MaximumBandwidth {serverinfo.bandwidth}MB");

ps2.AddScript($"start-vm '{serverinfo.servername}'");

ps2.Invoke();

string result\_ip = "获取中...";

//创建服务器后，添加记录到数据库

db.servers.Add(new Server()

{

cpu = serverinfo.cpu

,

disk = serverinfo.disk

,

ip = result\_ip

,

memory = serverinfo.memory

,

bandwidth = serverinfo.bandwidth

,

switchname = serverinfo.switchname

,

os = serverinfo.os

,

servername = serverinfo.servername

,

username = CurrentUser.Identity.Name

,

expire = DateTime.Now.AddMonths(months)

,

state = "Running"

});

db.SaveChanges();

db.transactions.Add(new Model.Transaction()

{

username = CurrentUser.Identity.Name,

type = "购买",

amount = cost,

time = DateTime.Now,

note = "服务器名称：" + serverinfo.servername

});

db.SaveChanges();

PowerShell ps3 = PowerShell.Create();

ps3.AddScript($"(get-vm '{serverinfo.servername}'| select -ExpandProperty networkadapters).ipAddresses");

int count = 180;

while (count != 0)

{

var o = ps3.Invoke();

if (o.Count > 1)

{

result\_ip = o[0].ToString();

var ss = db.servers.Where(s => s.servername == serverinfo.servername).FirstOrDefault();

ss.ip = result\_ip;

db.SaveChanges();

break;

}

Console.WriteLine("尝试：" + count);

await Task.Delay(1000);

count--;

}

message = "创建成功";

await show();

}

//向控制台输出创建的服务器相关信息

async Task show()

{

Console.WriteLine($"servername:{serverinfo.servername}");

Console.WriteLine($"VHDpath:{desVHDpath}");

Console.WriteLine($"path:{path + @"\" + serverinfo.servername}");

Console.WriteLine($"memory:{serverinfo.memory}");

Console.WriteLine($"cpu:{serverinfo.cpu}");

Console.WriteLine($"os:{serverinfo.os}");

Console.WriteLine($"disk:{serverinfo.disk}");

Console.WriteLine($"bandwidth:{serverinfo.bandwidth}");

var CurrentUser = (await authenticationStateTask).User;

Console.WriteLine($"username:{CurrentUser.Identity.Name}");

Console.WriteLine($"expire:{DateTime.Now.AddMonths(months)}");

}

}

**My server page: MyServer.razor**

@page "/myserver"

@inject CloudComputing.Data.ApplicationDbContext db

@inject AuthenticationStateProvider AuthenticationStateProvider

@using System.Management.Automation

@using System.Management.Automation.Runspaces

@attribute [Authorize]

<h3>我的服务器</h3>

<table class="table">

<tr>

<th>服务器名称</th>

<th>操作系统</th>

<th>内存大小</th>

<th>CPU</th>

<th>IP地址</th>

<th>带宽</th>

<th>交换机</th>

<th>硬盘大小</th>

<th>到期时间</th>

<th>状态</th>

<th>操作</th>

</tr>

@foreach (var server in servers)

{

<tr>

<td>@server.servername</td>

<td>@server.os</td>

<td>@server.memory MB</td>

<td>@server.cpu 核</td>

<td>@server.ip</td>

<td>@server.bandwidth MB</td>

<td>@server.switchname</td>

<td>@server.disk GB</td>

<td>@server.expire.ToString("yyyy-MM-dd")</td>

<td>@server.state</td>

<td>

<button class="btn btn-info btn-sm" **@onclick**="()=>Start(server)">启动</button>

<button class="btn btn-danger btn-sm" **@onclick**="()=>Stop(server)">停止</button>

<button class="btn btn-secondary btn-sm" **@onclick**="()=>Refresh(server)">刷新</button>

</td>

</tr>

}

</table>

@code {

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

//服务器列表

private List<Model.Server> servers;

//初始化

protected override async Task OnInitializedAsync()

{

var CurrentUser = (await authenticationStateTask).User;

servers = db.servers.Where(s => s.username == CurrentUser.Identity.Name).ToList();

}

//启动服务器

private void Start(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"Start-VM '{s.servername}'");

ps.Invoke();

s.state = "Running";

db.SaveChanges();

}

//停止服务器

private void Stop(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"Stop-VM '{s.servername}'");

ps.Invoke();

s.state = "Off";

db.SaveChanges();

}

//刷新服务器信息

private void Refresh(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"(get-vm '{s.servername}').state");

var result = ps.Invoke();

s.state = result[0].ToString();

db.SaveChanges();

//如果之前没有获取IP地址，则重新获取

if (s.ip== "获取中...")

{

PowerShell ps2 = PowerShell.Create();

ps2.AddScript($"(get-vm '{s.servername}' | select -ExpandProperty networkadapters).ipAddresses");

var r = ps2.Invoke();

s.ip=r[0].ToString();

db.SaveChanges();

}

}

}

**Transaction record page: UserTransaction.razor**

@page "/transaction"

@inject CloudComputing.Data.ApplicationDbContext db

@inject AuthenticationStateProvider AuthenticationStateProvider

@attribute [Authorize]

<h3>UserTransaction</h3>

<table class="table">

<tr>

<th>ID</th>

<th>操作</th>

<th>金额</th>

<th>发生时间</th>

<th>备注</th>

</tr>

@foreach (var transaction in transactions)

{

<tr>

<td>@transaction.id</td>

<td>@transaction.type</td>

<td>@transaction.amount</td>

<td>@transaction.time</td>

<td>@transaction.note</td>

</tr>

}

</table>

<div class="form-inline">

@if (pre == true)

{

<button class="btn btn-primary" **@onclick**="Pre">上一页</button>

}

@if (next == true)

{

<button class="btn btn-primary" **@onclick**="Next">下一页</button>

}

</div>

@code {

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

//交易记录列表

private List<Model.Transaction> transactions;

//上一页和下一页按钮默认为不显示

private bool pre = false, next = false;

//交易记录页码默认为1

private int pageindex = 1;

//交易记录一页最多显示20条

private int pagesize = 20;

//用于保存当前登录的用户名

private string username = "";

//初始化

protected override async Task OnInitializedAsync()

{

var CurrentUser = (await authenticationStateTask).User;

username = CurrentUser.Identity.Name;

if (db.transactions.Where(t => t.username == username).Count() > pagesize)

{

next = true;

}

transactions = db.transactions.Where(t => t.username == username).OrderByDescending(t => t.id).Take(20).ToList();

}

//下一页

private void Next()

{

pre = true;

pageindex++;

if (db.transactions.Count() > pagesize \* (pageindex))

{

next = true;

}

else

{

next = false;

}

transactions = db.transactions.Where(t => t.username == username).OrderByDescending(t => t.id).Skip(pagesize \* (pageindex - 1)).Take(20).ToList();

}

//上一页

private void Pre()

{

pageindex--;

if (pageindex == 1)

{

pre = false;

}

next = true;

transactions = db.transactions.Where(t => t.username == username).OrderByDescending(t => t.id).Skip(pagesize \* (pageindex - 1)).Take(20).ToList();

}

}

**Server management page(Administrator): ServerManager.razor**

@page "/servermanagement"

@inject CloudComputing.Data.ApplicationDbContext db

@using System.Management.Automation

@using System.Management.Automation.Runspaces

@attribute [Authorize(Roles = "Administrators")]

<h3>服务器管理</h3>

<table class="table">

<tr>

<th>用户名</th>

<th>服务器名称</th>

<th>操作系统</th>

<th>内存大小</th>

<th>CPU</th>

<th>IP地址</th>

<th>带宽</th>

<th>交换机</th>

<th>硬盘大小</th>

<th>到期时间</th>

<th>状态</th>

<th>操作</th>

</tr>

@foreach (var server in servers)

{

<tr>

<td>@server.username</td>

<td>@server.servername</td>

<td>@server.os</td>

<td>@server.memory MB</td>

<td>@server.cpu 核</td>

<td>@server.ip</td>

<td>@server.bandwidth MB</td>

<td>@server.switchname</td>

<td>@server.disk GB</td>

<td>@server.expire.ToString("yyyy-MM-dd")</td>

<td>@server.state</td>

<td>

<button class="btn btn-info btn-sm" **@onclick**="()=>Start(server)">启动</button>

<button class="btn btn-danger btn-sm" **@onclick**="()=>Stop(server)">停止</button>

<button class="btn btn-secondary btn-sm" **@onclick**="()=>Refresh(server)">刷新</button>

<button class="btn btn-dark btn-sm" **@onclick**="()=>Delete(server)">删除</button>

</td>

</tr>

}

</table>

@code {

//服务器列表

private List<Model.Server> servers;

//初始化

protected override void OnInitialized()

{

servers = db.servers.ToList();

}

//启动服务器

private void Start(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"Start-VM '{s.servername}'");

ps.Invoke();

s.state = "Running";

db.SaveChanges();

}

//停止服务器

private void Stop(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"Stop-VM '{s.servername}'");

ps.Invoke();

s.state = "Off";

db.SaveChanges();

}

//刷新服务器信息

private void Refresh(Model.Server s)

{

PowerShell ps = PowerShell.Create();

ps.AddScript($"(get-vm '{s.servername}').state");

var result = ps.Invoke();

s.state = result[0].ToString();

db.SaveChanges();

//如果之前没有获取IP地址，则重新获取

if (s.ip == "获取中...")

{

PowerShell ps2 = PowerShell.Create();

ps2.AddScript($"(get-vm '{s.servername}' | select -ExpandProperty networkadapters).ipAddresses");

var r = ps2.Invoke();

s.ip = r[0].ToString();

db.SaveChanges();

}

}

//删除服务器

private void Delete(Model.Server s)

{

PowerShell ps1 = PowerShell.Create();

ps1.AddScript($"Stop-VM '{s.servername}'");

ps1.Invoke();

PowerShell ps2 = PowerShell.Create();

ps2.AddScript($"Remove-VM '{s.servername}' -Force");

ps2.Invoke();

PowerShell ps3 = PowerShell.Create();

ps3.AddScript($@"Remove-Item 'C:\Users\dada\Desktop\Hyper-v\VM\{s.servername}' -Recurse -Force");

ps3.Invoke();

db.servers.Remove(s);

db.SaveChanges();

servers = db.servers.ToList();

}

}

**User management page(Administrator): UserManagement.razor**

@page "/usermanagement"

@using Microsoft.AspNetCore.Authorization

@using Microsoft.AspNetCore.Identity

@inject UserManager<IdentityUser> \_userManager

@inject RoleManager<IdentityRole> \_roleManager

@inject AuthenticationStateProvider AuthenticationStateProvider

<h3>Administration</h3>

<**AuthorizeView** **Roles**="Administrators">

<**Authorized**>

@if (context.User.IsInRole(ADMINISTRATION\_ROLE))

{

<table class="table">

<thead>

<tr>

<th>Id</th>

<th>用户名</th>

<th>邮箱</th>

</tr>

</thead>

<tbody>

@foreach (var user in ColUsers)

{

<tr>

<td>@user.Id.Substring(0, 5) ...</td>

<td>@user.UserName</td>

<td>@user.Email</td>

<td>

<!-- 编辑用户 -->

<button class="btn btn-primary"

**@onclick**="(() => EditUser(user))">

编辑

</button>

</td>

</tr>

}

</tbody>

</table>

}

else

{

<p>You're not signed in as a user in @ADMINISTRATION\_ROLE.</p>

}

@if (ShowPopup)

{

<!-- 弹窗创建用户或编辑用户 -->

<div class="modal" tabindex="-1" style="display:block" role="dialog">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h3 class="modal-title">用户信息</h3>

<!-- 关闭弹窗按钮 -->

<button type="button" class="close"

**@onclick**="ClosePopup">

<span aria-hidden="true">X</span>

</button>

</div>

<div class="modal-body">

@if (objUser.Id != "")

{

<p>@objUser.Id</p>

}

@if (objUser.Id != "")

{

<p>@objUser.UserName</p>

}

else

{

<input class="form-control" type="text"

placeholder="UserName"

**@bind**="objUser.UserName" />

}

<input class="form-control" type="text"

placeholder="Email"

**@bind**="objUser.Email" />

<input class="form-control" type="password"

placeholder="Password"

**@bind**="objUser.PasswordHash" />

<select class="form-control"

**@bind**="@CurrentUserRole">

@foreach (var option in Options)

{

<option value="@option">

@option

</option>

}

</select>

<br /><br />

<!-- 保存用户按钮 -->

<button class="btn btn-primary"

**@onclick**="SaveUser">

保存

</button>

@if (objUser.Id != "")

{

<!-- 删除用户按钮 -->

<button class="btn btn-danger"

**@onclick**="DeleteUser">

删除

</button>

}

<br />

<span style="color:red">@strError</span>

</div>

</div>

</div>

</div>

}

<button class="btn btn-info" **@onclick**="AddNewUser">添加新用户</button>

</**Authorized**>

<**NotAuthorized**>

<p>请先登录.</p>

</**NotAuthorized**>

</**AuthorizeView**>

@code {

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

string ADMINISTRATION\_ROLE = "Administrators";

System.Security.Claims.ClaimsPrincipal CurrentUser;

protected override async Task OnInitializedAsync()

{

// 确保是否有 ADMINISTRATION\_ROLE 角色

var RoleResult = await \_roleManager.FindByNameAsync(ADMINISTRATION\_ROLE);

if (RoleResult == null)

{

// 创建 ADMINISTRATION\_ROLE 角色

await \_roleManager.CreateAsync(new IdentityRole(ADMINISTRATION\_ROLE));

}

// 确保用户admin@1000.red是管理员

var user = await \_userManager.FindByNameAsync("admin@1000.red");

if (user != null)

{

// 角色admin@1000.red是否属于管理员?

var UserResult = await \_userManager.IsInRoleAsync(user, ADMINISTRATION\_ROLE);

if (!UserResult)

{

// 将用户添加到管理员角色

await \_userManager.AddToRoleAsync(user, ADMINISTRATION\_ROLE);

}

}

// 获取当前登录的用户

CurrentUser = (await authenticationStateTask).User;

GetUsers();

}

// 用户模型

IdentityUser objUser = new IdentityUser();

// 选中用户的角色

string CurrentUserRole { get; set; } = "Users";

// 用户集合

List<IdentityUser> ColUsers = new List<IdentityUser>();

// 可选角色

List<string> Options = new List<string>() { "Users", "Administrators" };

// 错误信息

string strError = "";

// 是否显示弹窗

bool ShowPopup = false;

void AddNewUser()

{

// Make new user

objUser = new IdentityUser();

objUser.PasswordHash = "\*\*\*\*\*";

// 将id设置为空，表示其为新记录

objUser.Id = "";

// 打开弹窗

ShowPopup = true;

}

async Task SaveUser()

{

try

{

if (objUser.Id != "")

{

// 获取用户

var user = await \_userManager.FindByIdAsync(objUser.Id);

// 更新邮箱

user.Email = objUser.Email;

// 更新用户

await \_userManager.UpdateAsync(user);

// 只有当密码不是默认密码时才进行更新

if (objUser.PasswordHash != "\*\*\*\*\*")

{

var resetToken =

await \_userManager.GeneratePasswordResetTokenAsync(user);

var passworduser =

await \_userManager.ResetPasswordAsync(

user,

resetToken,

objUser.PasswordHash);

if (!passworduser.Succeeded)

{

if (passworduser.Errors.FirstOrDefault() != null)

{

strError =

passworduser

.Errors

.FirstOrDefault()

.Description;

}

else

{

strError = "Pasword error";

}

// 保持弹窗打开

return;

}

}

// 用户是否为管理员角色

var UserResult =

await \_userManager

.IsInRoleAsync(user, ADMINISTRATION\_ROLE);

// 选择了管理员但是用户不是管理员

if (

(CurrentUserRole == ADMINISTRATION\_ROLE)

&

(!UserResult))

{

// 将用户添加到管理员角色

await \_userManager

.AddToRoleAsync(user, ADMINISTRATION\_ROLE);

}

else

{

// 没有选择管理员但用户是管理员

if ((CurrentUserRole != ADMINISTRATION\_ROLE)

&

(UserResult))

{

// 将用户移除管理员角色

await \_userManager

.RemoveFromRoleAsync(user, ADMINISTRATION\_ROLE);

}

}

}

else

{

// 添加新用户

var NewUser =

new IdentityUser

{

UserName = objUser.UserName,

Email = objUser.Email

};

var CreateResult =

await \_userManager

.CreateAsync(NewUser, objUser.PasswordHash);

if (!CreateResult.Succeeded)

{

if (CreateResult

.Errors

.FirstOrDefault() != null)

{

strError =

CreateResult

.Errors

.FirstOrDefault()

.Description;

}

else

{

strError = "Create error";

}

// 保持弹窗打开

return;

}

else

{

if (CurrentUserRole == ADMINISTRATION\_ROLE)

{

// 将用户添加到管理员角色

await \_userManager

.AddToRoleAsync(NewUser, ADMINISTRATION\_ROLE);

}

}

}

// 关闭弹窗

ShowPopup = false;

// 刷新用户列表

GetUsers();

}

catch (Exception ex)

{

strError = ex.GetBaseException().Message;

}

}

async Task EditUser(IdentityUser \_IdentityUser)

{

// 选择用户

objUser = \_IdentityUser;

// 获取用户

var user = await \_userManager.FindByIdAsync(objUser.Id);

if (user != null)

{

// 用户是否属于管理员角色

var UserResult =

await \_userManager

.IsInRoleAsync(user, ADMINISTRATION\_ROLE);

if (UserResult)

{

CurrentUserRole = ADMINISTRATION\_ROLE;

}

else

{

CurrentUserRole = "Users";

}

}

// 打开弹窗

ShowPopup = true;

}

async Task DeleteUser()

{

// 关闭弹窗

ShowPopup = false;

// 获取用户

var user = await \_userManager.FindByIdAsync(objUser.Id);

if (user != null)

{

// 删除用户

await \_userManager.DeleteAsync(user);

}

// 刷新用户列表

GetUsers();

}

void ClosePopup()

{

// 关闭弹窗

ShowPopup = false;

}

public void GetUsers()

{

// 清除错误信息

strError = "";

// 保存用户的集合

ColUsers = new List<IdentityUser>();

// 从 \_UserManager 获取用户

var user = \_userManager.Users.Select(x => new IdentityUser

{

Id = x.Id,

UserName = x.UserName,

Email = x.Email,

PasswordHash = "\*\*\*\*\*"

});

foreach (var item in user)

{

ColUsers.Add(item);

}

}

}

**Template management(Administrator): TemplateManagement.razor**

@page "/templatemanagement"

@inject CloudComputing.Data.ApplicationDbContext db

@attribute [Authorize]

@using CloudComputing.Model

<h3>Template Management</h3>

<**AuthorizeView** **Roles**="Administrators">

<div class="form-inline">

<div class="form-group">

<label>系统名称：</label>

<input type="text" class="form-control" **@bind-value**="ost.name" />

</div>

<div class="ml-3 form-inline">

<label>镜像地址：</label>

<input type="text" class="form-control" **@bind-value**="ost.path" />

</div>

<div class="ml-3 form-inline">

<label>默认密码：</label>

<input type="text" class="form-control" **@bind-value**="ost.password" />

</div>

<button class="btn btn-primary" **@onclick**="Add">添加</button>

@if (update)

{

<button class="btn btn-success" **@onclick**="Update">更新</button>

<button class="btn btn-danger" **@onclick**="Cancel">取消</button>

}

</div>

<table class="table">

<tr>

<th>系统名称</th>

<th>镜像地址</th>

<th>默认密码</th>

<th>操作</th>

</tr>

@foreach (var template in templates)

{

<tr>

<td>@template.name</td>

<td>@template.path</td>

<td>@template.password</td>

<td>

<button class="btn btn-info btn-sm" **@onclick**="()=>Edit(template)">编辑</button>

<button class="btn btn-dark btn-sm" **@onclick**="()=>Delete(template)">删除</button>

</td>

</tr>

}

</table>

</**AuthorizeView**>

@code {

//是否显示更新相关按钮

private bool update = false;

//页面系统镜像模板信息模型

private Template ost = new Template() { name = "", path = "" };

//系统镜像模板列表

private List<Template> templates;

//初始化

protected override async Task OnInitializedAsync()

{

templates = db.templates.ToList();

}

//添加新系统镜像模板

private void Add()

{

db.templates.Add(new Template()

{

name = ost.name,

path = ost.path,

password=ost.password

});

db.SaveChanges();

templates = db.templates.ToList();

Cancel();

}

//编辑系统镜像模板

private void Edit(Template t)

{

ost = t;

update = true;

}

//删除系统镜像模板

private void Delete(Template t)

{

db.templates.Remove(t);

db.SaveChanges();

templates = db.templates.ToList();

}

//更新系统镜像模板

private void Update()

{

db.templates.Update(ost);

db.SaveChanges();

templates = db.templates.ToList();

Cancel();

}

//取消更新系统镜像模板

private void Cancel()

{

ost = new Template() { name = "", path = "", password = "" };

update = false;

}

}

**Switch management(Administrator): SwitchManagement.razor**

@page "/switchmanagement"

@inject CloudComputing.Data.ApplicationDbContext db

@attribute [Authorize]

@using CloudComputing.Model

<h3>SwitchManagement</h3>

<**AuthorizeView** **Roles**="Administrators">

<div class="form-inline">

<div class="form-group">

<label>网络名称：</label>

<input type="text" class="form-control" **@bind-value**="\_switch.name" />

</div>

<div class="ml-3 form-inline">

<label>虚拟交换机：</label>

<input type="text" class="form-control" **@bind-value**="\_switch.virtualswitch" />

</div>

<button class="btn btn-primary" **@onclick**="Add">添加</button>

@if (update)

{

<button class="btn btn-success" **@onclick**="Update">更新</button>

<button class="btn btn-danger" **@onclick**="Cancel">取消</button>

}

</div>

<table class="table">

<tr>

<th>网络名称</th>

<th>虚拟交换机</th>

<th>操作</th>

</tr>

@foreach (var s in switches)

{

<tr>

<td>@s.name</td>

<td>@s.virtualswitch</td>

<td>

<button class="btn btn-info btn-sm" **@onclick**="()=>Edit(s)">编辑</button>

<button class="btn btn-dark btn-sm" **@onclick**="()=>Delete(s)">删除</button>

</td>

</tr>

}

</table>

</**AuthorizeView**>

@code {

//是否显示更新相关按钮

private bool update = false;

//页面交换机信息模型

private Switch \_switch = new Switch() { name = "", virtualswitch = "" };

//交换机列表

private List<Switch> switches;

//初始化

protected override async Task OnInitializedAsync()

{

switches = db.switches.ToList();

}

//添加新交换机模板

private void Add()

{

db.switches.Add(new Switch()

{

name = \_switch.name,

virtualswitch = \_switch.virtualswitch

});

db.SaveChanges();

switches = db.switches.ToList();

Cancel();

}

//编辑交换机模板

private void Edit(Switch s)

{

\_switch = s;

update = true;

}

//删除交换机模板

private void Delete(Switch s)

{

db.switches.Remove(s);

db.SaveChanges();

switches = db.switches.ToList();

}

//更新交换机模板

private void Update()

{

db.switches.Update(\_switch);

db.SaveChanges();

switches = db.switches.ToList();

Cancel();

}

//取消更新操作

private void Cancel()

{

\_switch = new Switch() { name = "", virtualswitch = ""};

update = false;

}

}

**All transaction record(Administrator): AllTransaction.razor**

@page "/alltransaction"

@inject CloudComputing.Data.ApplicationDbContext db

@inject AuthenticationStateProvider AuthenticationStateProvider

@attribute [Authorize]

<**AuthorizeView** **Roles**="Administrators">

<div class="form-inline">

<div class="form-group">

<label>用户名：</label>

<input type="text" class="form-control" **@bind-value**="username" />

</div>

<button class="btn btn-primary" **@onclick**="GetUserTransaction">查找</button>

<div class="ml-3 form-inline">

操作类型：<select **@bind**="type" class="form-control">

<option>未选择</option>

<option>充值</option>

<option>购买</option>

</select>

</div>

<button class="btn btn-primary" **@onclick**="FilterByType">过滤</button>

</div>

<table class="table">

<tr>

<th>ID</th>

<th>用户名</th>

<th>操作</th>

<th>金额</th>

<th>发生时间</th>

<th>备注</th>

</tr>

@foreach (var transaction in transactions)

{

<tr>

<td>@transaction.id</td>

<td>@transaction.username</td>

<td>@transaction.type</td>

<td>@transaction.amount</td>

<td>@transaction.time</td>

<td>@transaction.note</td>

</tr>

}

</table>

<div class="form-inline">

@if (pre == true)

{

<button class="btn btn-primary" **@onclick**="Pre">上一页</button>

}

@if(next == true)

{

<button class="btn btn-primary" **@onclick**="Next">下一页</button>

}

</div>

</**AuthorizeView**>

@code {

[CascadingParameter]

private Task<AuthenticationState> authenticationStateTask { get; set; }

//交易记录列表

private List<Model.Transaction> transactions;

//上一页和下一页按钮默认为不显示

private bool pre = false, next = false;

//交易记录默认页码为1

private int pageindex = 1;

//一页最多显示20条交易记录

private int pagesize = 20;

//交易类型

private string type { get; set; }

//用户名

private string username = "";

//初始化

protected override async Task OnInitializedAsync()

{

if(db.transactions.Count()>pagesize)

{

next = true;

}

transactions = db.transactions.OrderByDescending(t => t.id).Take(20).ToList();

}

//获取用户交易记录

private void GetUserTransaction()

{

if (username == "")

{

transactions = db.transactions.OrderByDescending(t => t.id).Take(20).ToList();

}

else

{

transactions = db.transactions.Where(t => t.username == username).OrderByDescending(t => t.id).Take(20).ToList();

}

}

//通过交易类型过滤

private void FilterByType()

{

if (username == "")

{

transactions = db.transactions.OrderByDescending(t => t.id).Take(20).ToList();

}

else

{

transactions = db.transactions.Where(t => t.username == username).OrderByDescending(t => t.id).Take(20).ToList();

}

if (type == "未选择")

{

return;

}

if (type == "充值")

{

transactions = transactions.Where(t => t.type == "充值").ToList();

}

if (type == "购买")

{

transactions = transactions.Where(t => t.type == "购买").ToList();

}

}

//下一页

private void Next()

{

pre = true;

pageindex++;

if (db.transactions.Count() > pagesize\*(pageindex))

{

next = true;

}

else

{

next = false;

}

transactions = db.transactions.OrderByDescending(t => t.id).Skip(pagesize\*(pageindex-1)).Take(20).ToList();

}

//上一页

private void Pre()

{

pageindex--;

if (pageindex==1)

{

pre = false;

}

next = true;

transactions = db.transactions.OrderByDescending(t => t.id).Skip(pagesize \* (pageindex - 1)).Take(20).ToList();

}

}

## 8.4 Dependency injection and middleware configuration

**Project configuration file: Startup.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Components;

using Microsoft.AspNetCore.Components.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.UI;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using CloudComputing.Areas.Identity;

using CloudComputing.Data;

using Microsoft.Extensions.Options;

namespace CloudComputing

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(

Configuration.GetConnectionString("DefaultConnection")));

services.AddDefaultIdentity<IdentityUser>(options => {

options.SignIn.RequireConfirmedAccount = false;

options.Password.RequireNonAlphanumeric = false; // Do not require special characters

}).AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

services.AddHttpContextAccessor(); // get current username

services.AddRazorPages();

services.AddServerSideBlazor();

services.AddScoped<AuthenticationStateProvider, RevalidatingIdentityAuthenticationStateProvider<IdentityUser>>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseDatabaseErrorPage();

}

else

{

app.UseExceptionHandler("/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

endpoints.MapBlazorHub();

endpoints.MapFallbackToPage("/\_Host");

});

}

}

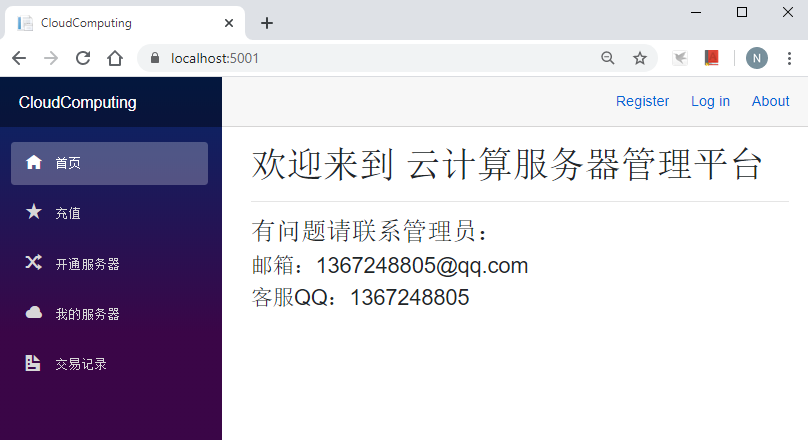
}

# 9. System running analysis

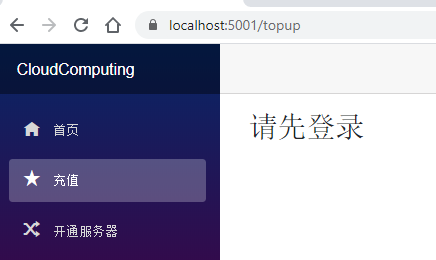
## 9.1 Not logged in

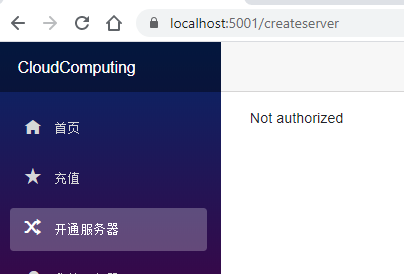
This system has designed the authorization authentication system, and achieved the effect that different user role login can display the different page content. In addition, the specific page also has a separate authentication, which can be used to determine whether the user is logged in and whether the user is an administrator. Therefore, our system is very secure. Even manually entering the URL address to access the administrator's exclusive management page cannot be authorized.

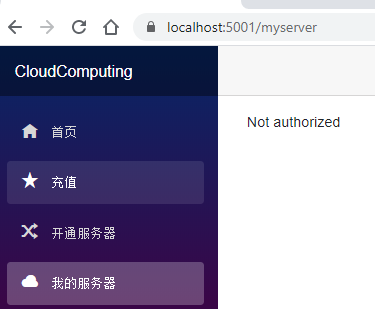
Default homepage when not logged in:

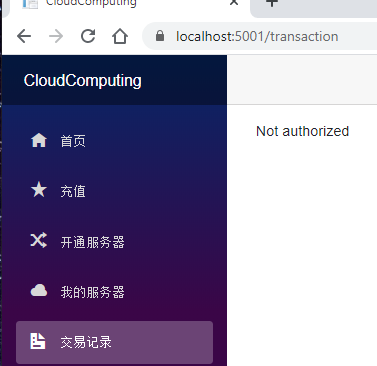


When we try to access different menu item:



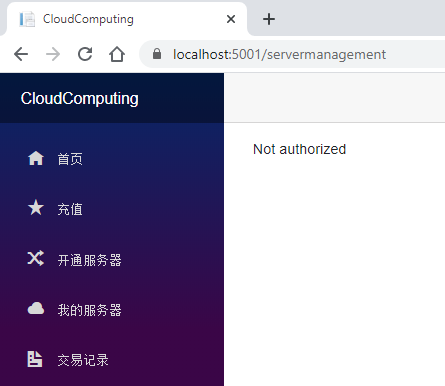






We can find that all pages cannot be accessed when we not logged in.

We can manually type administrator’s [Server Management] URL address:



Also cannot be accessed, our system is very secure.

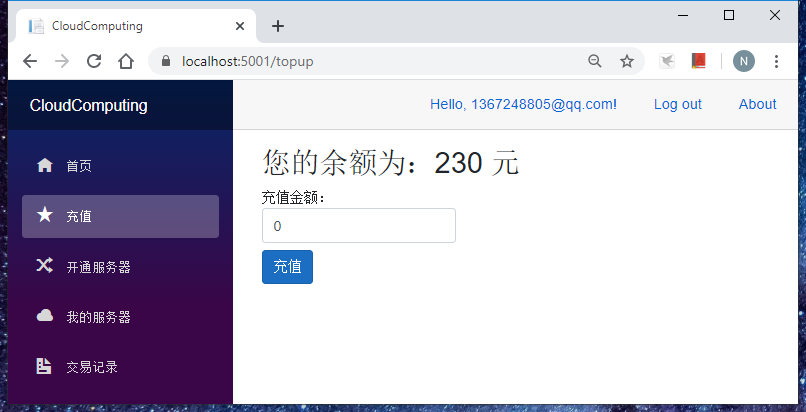
## 9.2 Logged in as user

Homepage when logged in as user:

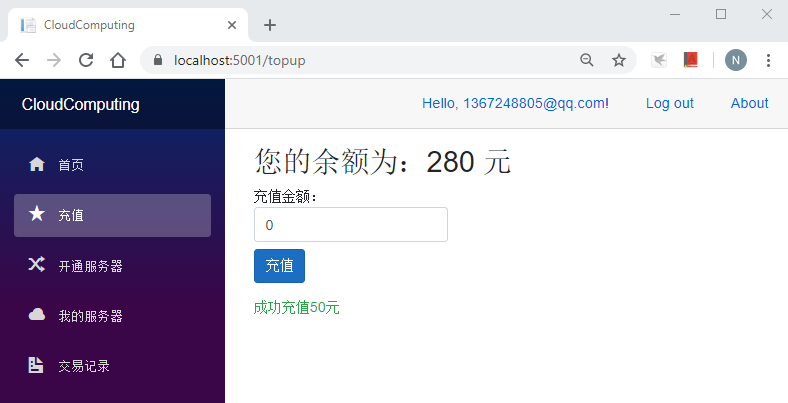


Normal user only has [Top Up], [Create Server], [My Server], [Transaction] menu items.

Choose [Top Up] menu item:

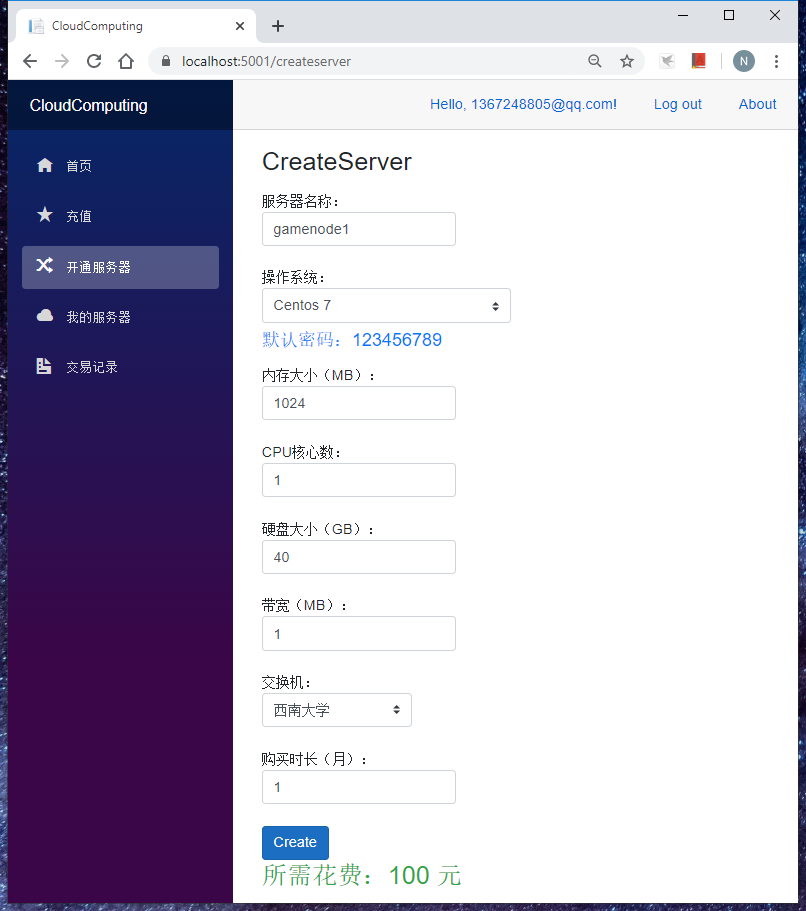


We can try to top up 50 Yuan.



Then we can try to create a new server

our server name could be: gamenode1



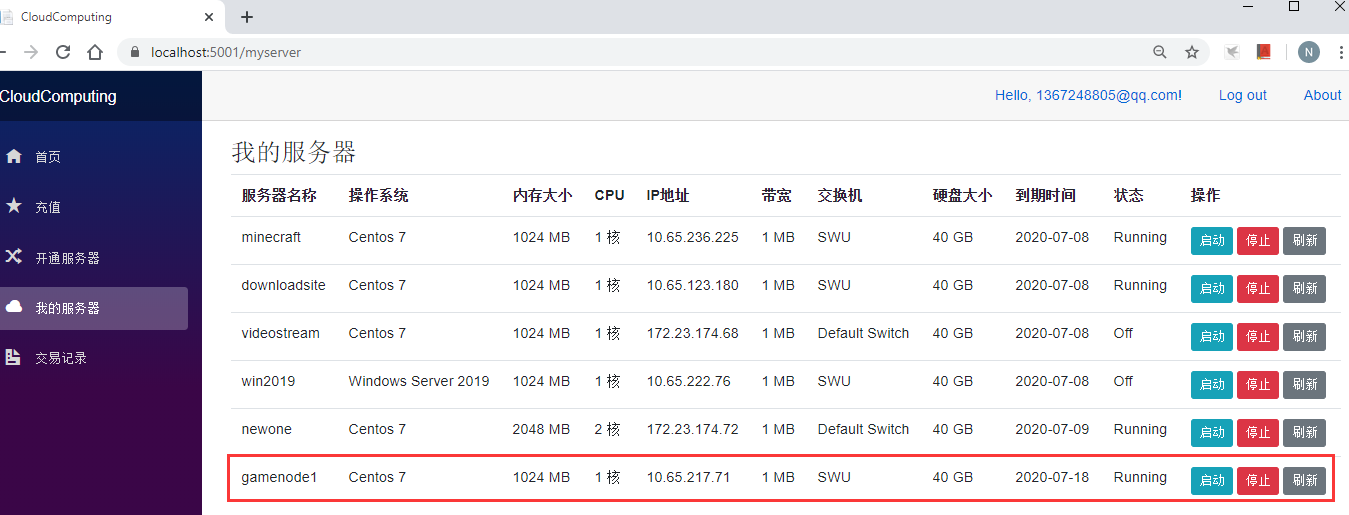
After we clicked “Create” button, system will tell us now is creating:

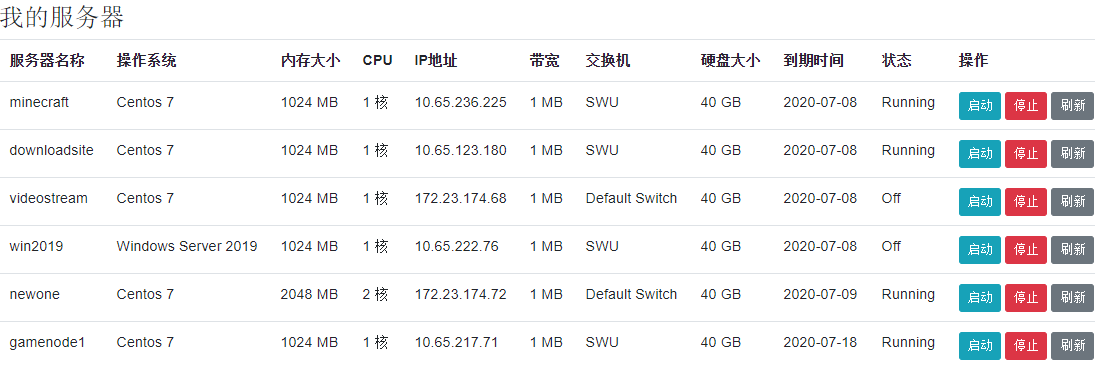


After created completely, system will tell us now is finished.



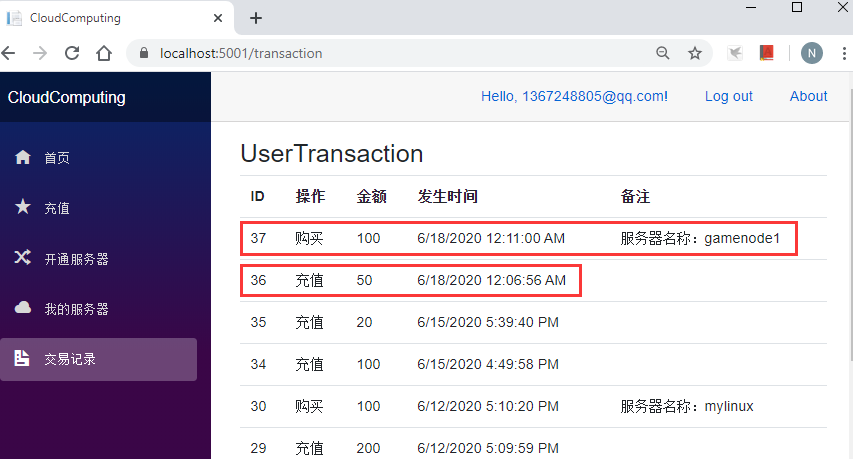
After created our server, we can check our server in [My Server] menu item:





We can manage our server at this page, such as start or stop.

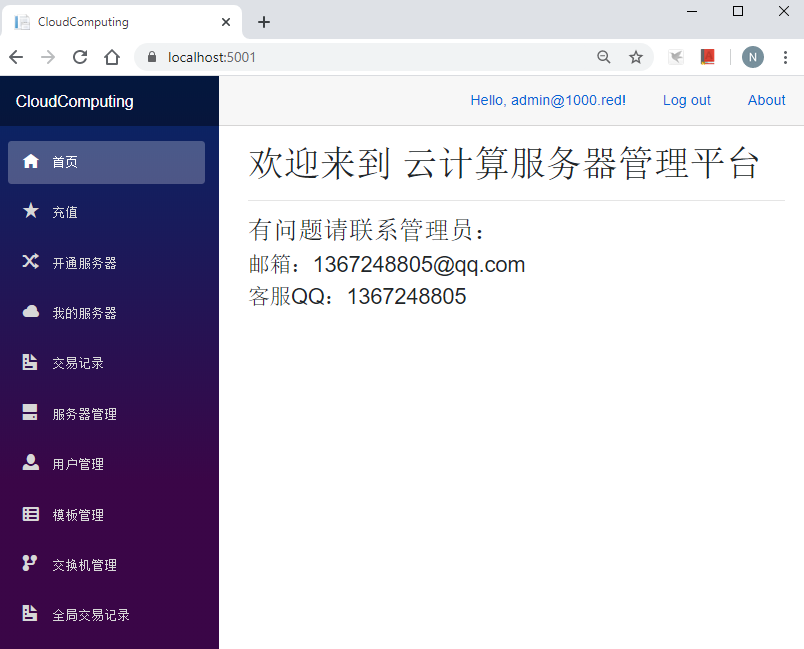
Finally, we can click [Transaction] to check our records:



We can see the server “gamenode1” we just created, and the 50 Yuan we just top up.

## 9.3 Logged in as administrator

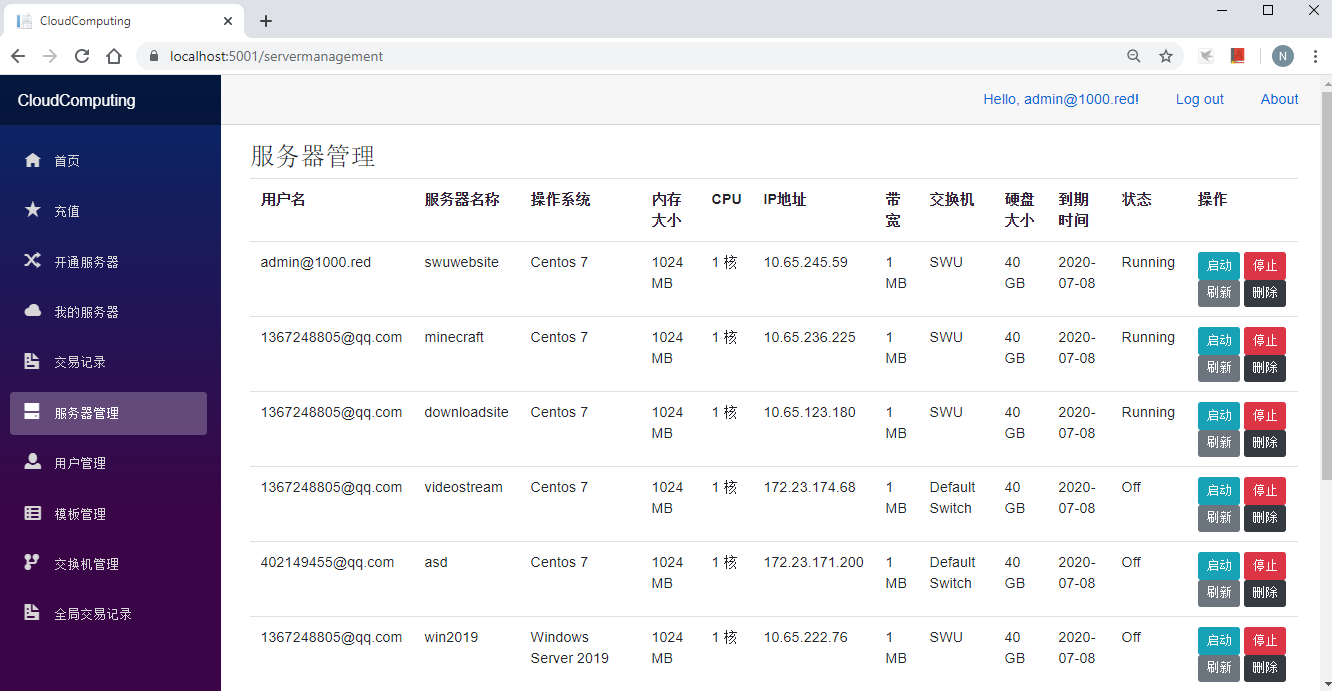
Homepage when logged in as administrator:



Administrator not only has [Top Up], [Create Server], [My Server], [Transaction] menu items, but also has [Server Management], [User Management], [Template Management], [Switch Management], [All Transaction] menu items.

Administrator’s [Top Up], [Create Server], [My Server], [Transaction] menu items are the same as users’.

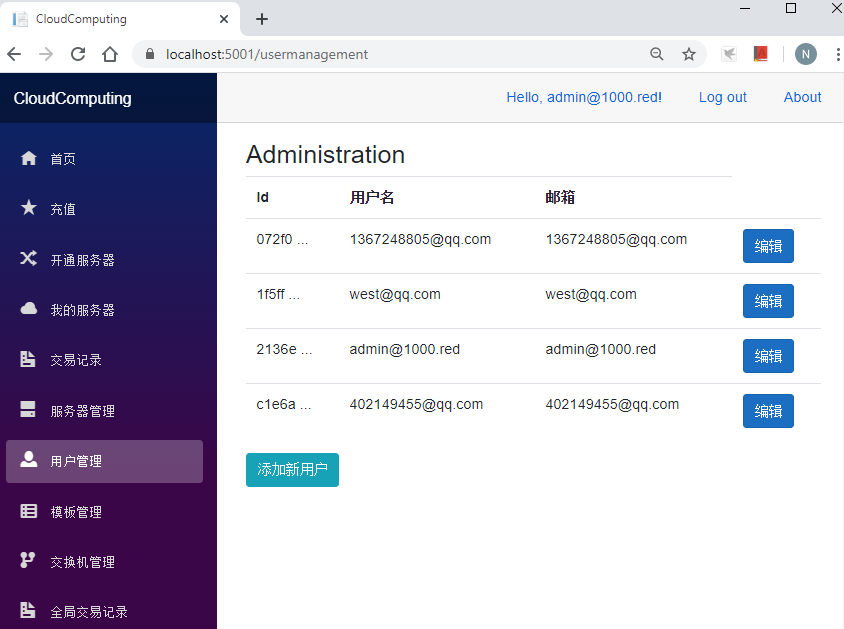
Administrator can manage all users’ server; we click [Server Management]:



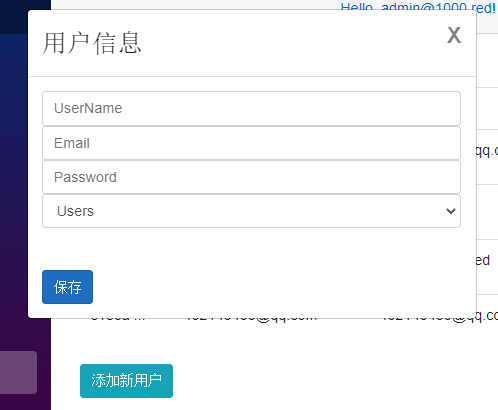


Administrator’s [Server Management] is similar to [My Server] menu item, however added Username field to indicate the server’s owner, and added the ‘delete’ option to let administrator can delete specific server.

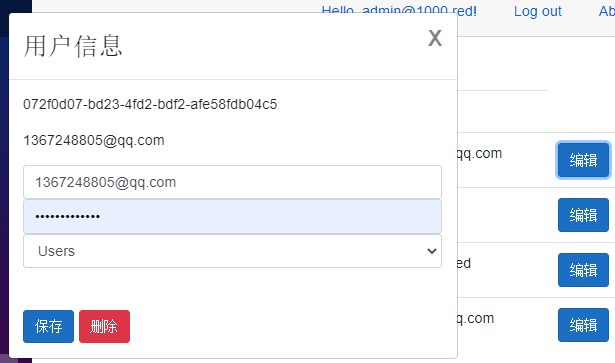
[User Management] menu can let administrator add, edit or delete user:



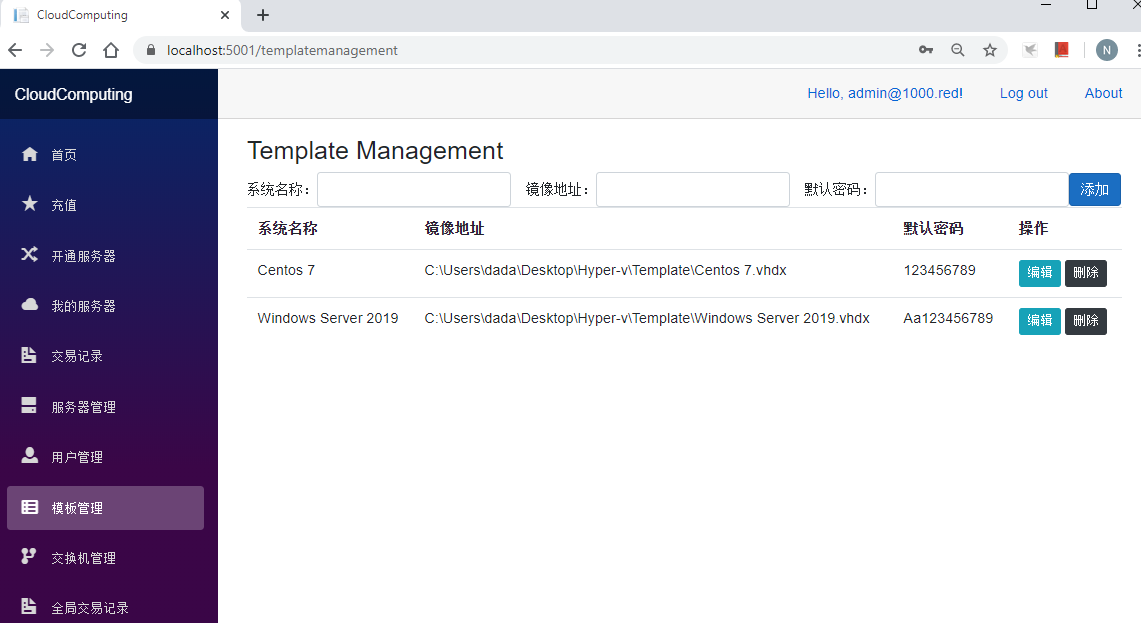
Click “add new user” button can add customized user information.



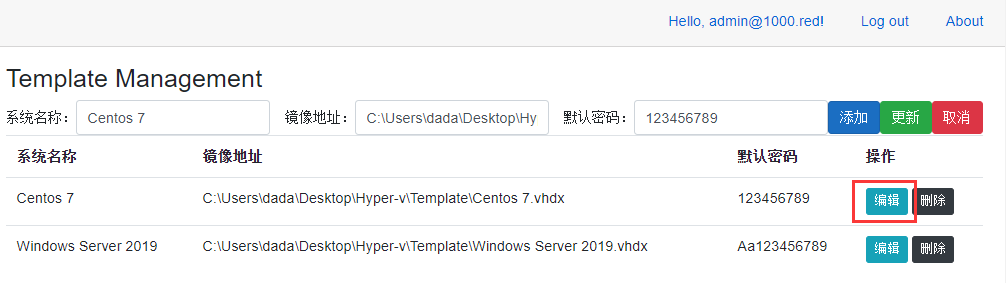
After click “edit” button administrator can modify user’s information or delete user:



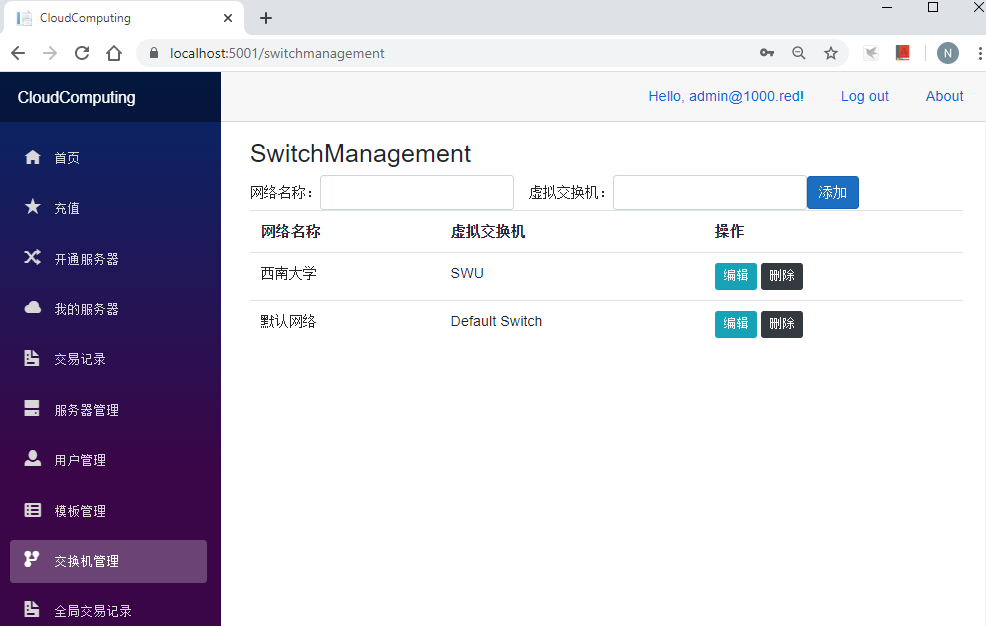
[Template Management] menu item can manage operating system templates which are available when creating a new server:



Administrator can add, edit or delete templates:



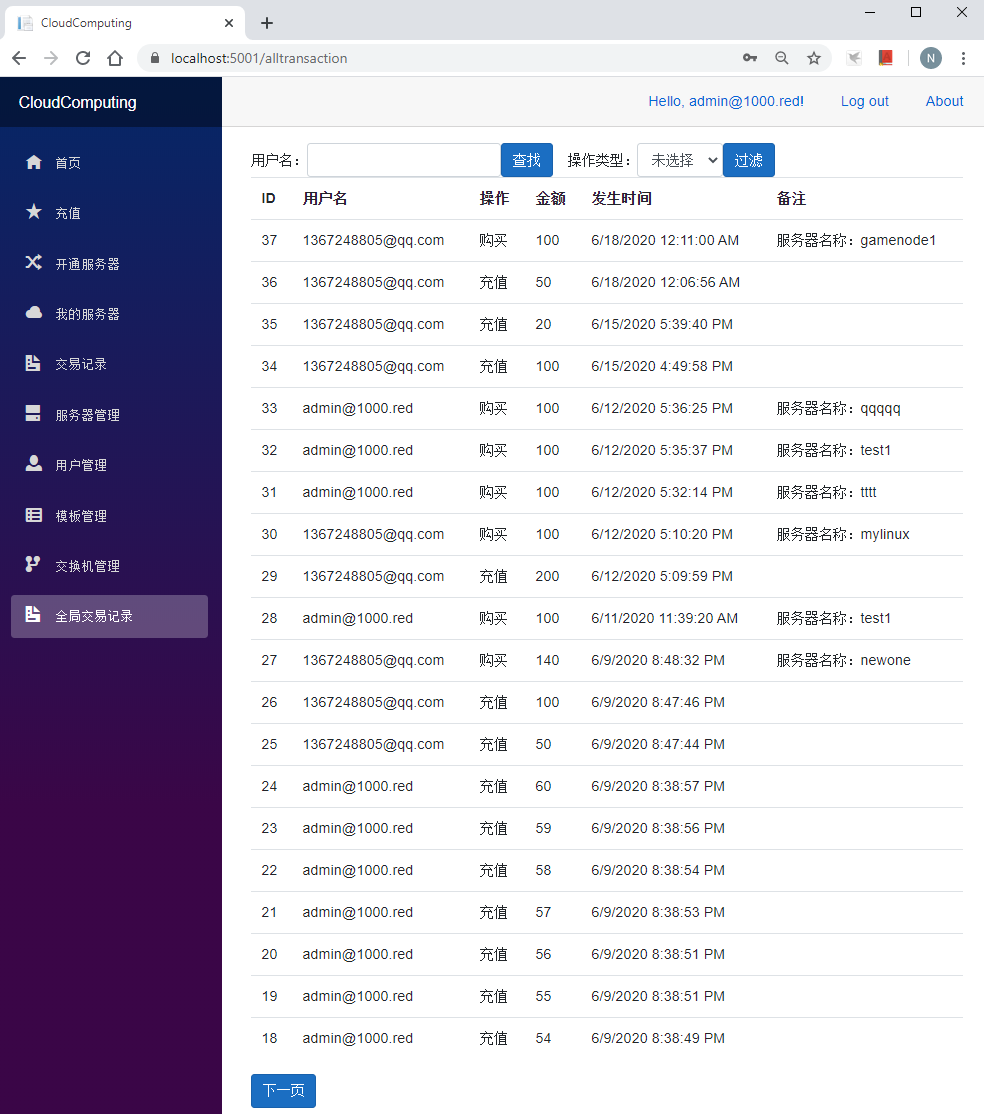
[Switch Management] can manage switch templates which are available when creating a new server:



Administrator can add, edit or delete virtual switch templates:



[All Transaction] menu item can let administrator check every users’ transaction records:



Administrator can search specific user’s transaction records, or filter records by operation type, and we can click “next page”, “previous page” to check more records.

# 10. Summary

Security:

As a web site system, the number and behavior of users are unpredictable. We must pay attention to the authorization and authentication system of the website, improve the security of the system, and ensure that the system is not damaged by hackers and illegally used by lawbreakers. This time, I added the authorization and authentication check to each individual page to ensure that every page will be safe.

Abstract layer of database operation:

This time, I adopted the Entity Framework core framework as the abstract layer of data access operations. The advantage of using the abstract layer is that our development is separated from the specific underlying database software to make our system more applicable. In the future, when other people use our system, they can choose their favorite database programs without worrying that our system only supports a specific database program. This greatly improves the flexibility and convenience of our system deployment.

Interface design:

The traditional website interface style is not uniform. After entering different functions, the page layout changes, which greatly increases the user's use and learning costs. I adopted a very simple page layout design this time, with all the function menu options on the left. When users switch between different functions, the menu items will not change, which greatly reduces the user's use difficulty, makes users easier to use, and improves user satisfaction.

Generality:

Sometimes, in order to implement some functions quickly, the values of some variables will be written in the code directly. Therefore, when other people want to make some changes, there is no way to do so. In order to improve our system’s flexibilities, I added many setting pages to the system this time, which can allow the administrator to dynamically modify the system configuration and parameters. Without modifying the source code of the program, the system can meet the new change requirements. Make our system more practical and versatile.

Search and filter function:

I added the functions of searching by user name and filtering by operation type to the global transaction record function, which greatly improved the efficiency of the administrator to query transaction records. And the filtering function can separate the recharge transaction and the transaction of creating server at one time, which is convenient for the administrator to filter the transaction records that he wants to check.

# 11. References

Alibaba Cloud. (n.d.). Retrieved from https://www.aliyun.com/

Visual Studio IDE, Code Editor, Azure DevOps, & App Center. (2020, May 21). Retrieved

from https://visualstudio.microsoft.com/

Bootstrap. (2020). Retrieved 27 March 2020, from https://getbootstrap.com/

.NET: Free. Cross-platform. Open Source. (n.d.). Retrieved from https://dotnet.microsoft.com/

Blazor: Build client web apps with C#: .NET. (n.d.). Retrieved

from https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

Technical documentation, API, and code examples. (2020). Retrieved 16 June 2020, from https://docs.microsoft.com/