数据库实现报告

实现环境与主要技术

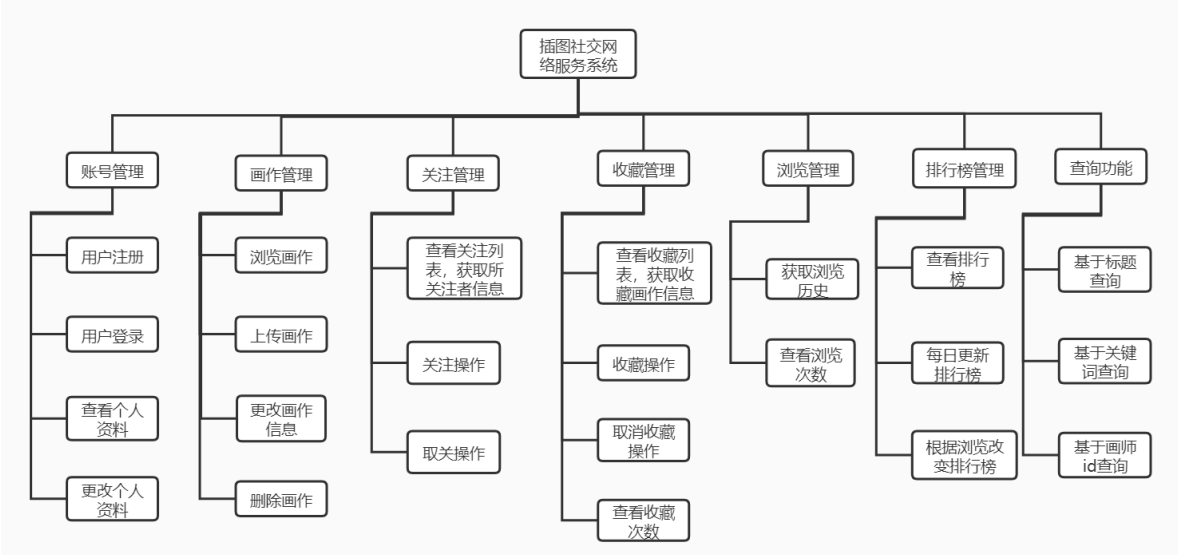
实现环境

操作系统： windows10
软件环境： jdk1.8+tomcat 9.0.27+ springboot 2.2.1+ sqlserver 2019+ vue 2.5.2+ webpack 3.6.0
开发环境：IntellijJ-IDEA+SSMS

主要技术

技术名称	功能
maven	包管理
git	版本控制
sqlserver	数据库
java	实现业务
JDBC	java与数据库连接
tomcat	开源servlet容器
spring-framework	基于IOC与AOP的系统框架
spring-boot	自动配置
html、css、JavaScript	页面设计与脚本控制
vue	渐进式前端框架

系统功能结构图



基本表/完整性约束/索引定义

基本表与完整性约束定义

```
CREATE TABLE users
(
   userID int identity(1,1) primary key,
   login_name varchar(16) NOT NULL UNIQUE,
   userName varchar(16) DEFAULT '没有名字(⊙__⊙)',
   pass_word varchar(16) NOT NULL,
   sex varchar(4) default '?' check(sex = '男' or sex='女' or sex='?'),
   address varchar(20) default '火星',
   profile_picture varchar(128) DEFAULT 'static/profilePicture/2.jpg',
   signature text NULL default '这个人很懒，什么都没写...'
);
go


create table painters(
    painter_id int PRIMARY KEY REFERENCES users(userID)  ,
    registeTime datetime default getdate(),
);
go


create table follows(
    follower_id int REFERENCES users(userID)  ,
    painter_id int REFERENCES painters(painter_id)  ,
    followTime datetime default getdate(),
);
go


create table pictures(
    picture_id int PRIMARY KEY,
    painter_id int REFERENCES painters(painter_id)  ,
    picture_address varchar(128) NOT NULL,
    uploadTime datetime default getdate(),
    title varchar(16) NOT NULL
);
go


create table keywords(
    picture_id int REFERENCES pictures(picture_id)  ,
    keyword varchar(16) NOT NULL
);
go


create table pictures_sets(
    pictures_sets_id int PRIMARY KEY,
    painter_id int REFERENCES painters(painter_id)  ,
    set_name varchar(16) NOT NULL,
    createTime datetime NOT NULL,
    remarks text,
    cover varchar(128) NOT NULL
```

```
);
go


create table pictures_sets_contents(
    pictures_sets_id int REFERENCES pictures_sets(pictures_sets_id)  ,
    picture_id int REFERENCES pictures(picture_id)
);
go


create table comments(
    comment_id int identity(1,1) PRIMARY KEY,
    commentator_id int REFERENCES users(userID)  ,
    picture_id int REFERENCES pictures(picture_id)  ,
    content text NOT NULL,
    commentTime datetime default getdate()
);
go


create table comment_likes(
    comment_id int REFERENCES comments(comment_id)  ,
    liker_id int REFERENCES users(userID)  ,
);
go


create table favorites(
    userID int REFERENCES users(userID)  ,
    favorites_name varchar(16) NOT NULL,
    createTime datetime NOT NULL,
    favorites_id int identity(1,1) PRIMARY KEY
);
go


create table favorites_pictures(
    favorites_id int REFERENCES favorites(favorites_id)  ,
    picture_id int REFERENCES pictures(picture_id)  ,
    collectTime datetime NOT NULL
);
go

create table browses(
    browse_id int identity(1,1) PRIMARY KEY,
    picture_id int REFERENCES pictures(picture_id)  ,
    browser_id int REFERENCES users(userID)  ,
    browseTime datetime default getdate(),
);
go


create table supports(
    support_id int PRIMARY KEY,
    sponsor_id int REFERENCES users(userID)  ,
    receiver_id int REFERENCES painters(painter_id)  ,
    amount float check(amount > 0),
```

```
    sponsorTime datetime NOT NULL
);
go


create table private_letters(
    private_letter_id int PRIMARY KEY,
    sender_id int REFERENCES users(userID)  ,
    receiver_id int REFERENCES users(userID)  ,
    content text NOT NULL,
    sentTime dateTime NOT NULL
);
go

create table collections(
    user_id int REFERENCES users(userID),
    picture_id int REFERENCES pictures(picture_id)  ,
    collectTime dateTime default getdate()
);
go
```

索引定义

```
--，建立关键字索引
create clustered index keywords_index on keywords(keyword)

--创建图片关于画师id的索引
create nonclustered index picture_index on pictures(painter_id)
```

系统安全性设计

用户创建时会产生属于自己的用户标识`user_id`，并采用静态口令鉴别方法进行用户身份鉴别。

为不同用户提供不同的数据库访问路径，如为画师只提供对自己的画作、以及用户对自己评论的管理路径。

不同用户会有不同的外模式与权限。如只有画师能对画作基本表进行增删改的操作，而普通用户不行。

存储过程、触发器与函数代码说明

```
--级联删除触发器
create trigger delete_comment on comments
instead of delete
as
begin
    --游标操作
    declare @comment_id int;
    declare tempCursor CURSOR LOCAL FOR (select comment_id from deleted);
    open tempCursor;
    fetch next from tempCursor into @comment_id
    while @@FETCH_STATUS=0
    begin
        print('正在删除...');
        print(@comment_id);
        delete from comment_likes where comment_id = @comment_id;
```

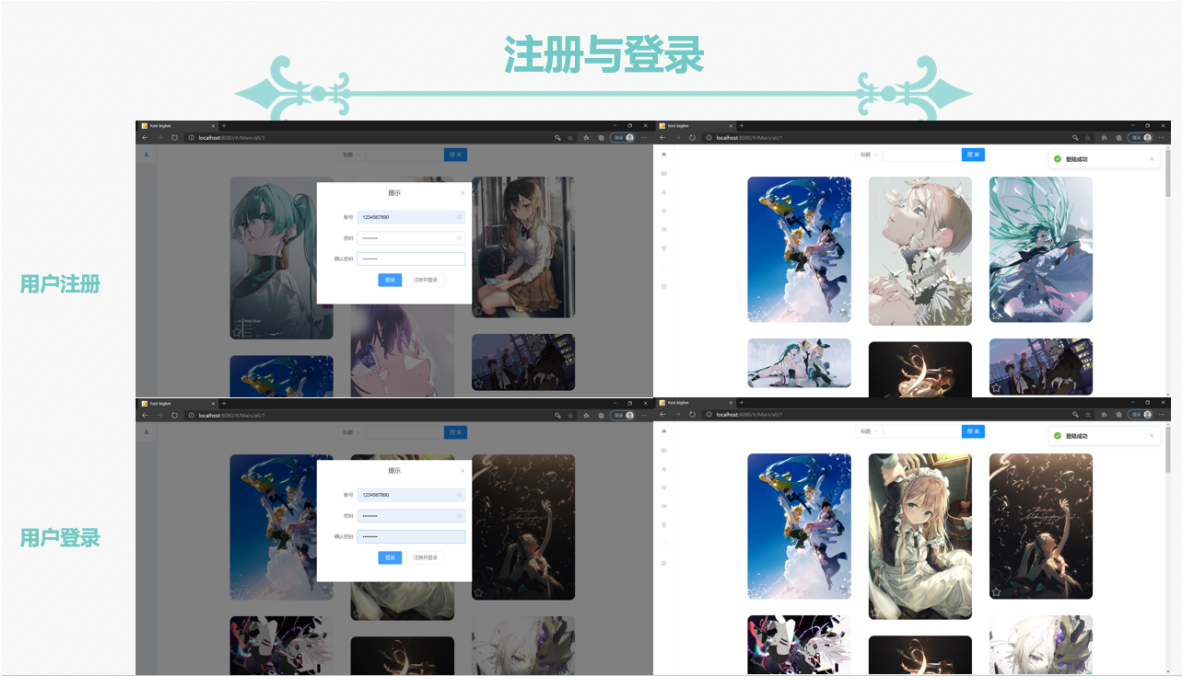
```
        delete from comments where comment_id = @comment_id;
        fetch next from tempCursor into @comment_id;
    end

end
go

--级联+嵌套 删除触发器
create trigger delete_picture on pictures
instead of delete
as
begin
    declare @picture_id int;
    select @picture_id = picture_id from deleted;
    delete from keywords where picture_id = @picture_id;
    delete from comments where picture_id = @picture_id;
    delete from browses where picture_id = @picture_id;
    delete from collections where picture_id = @picture_id;
    delete from pictures where picture_id = @picture_id;
end

```

系统功能运行实例

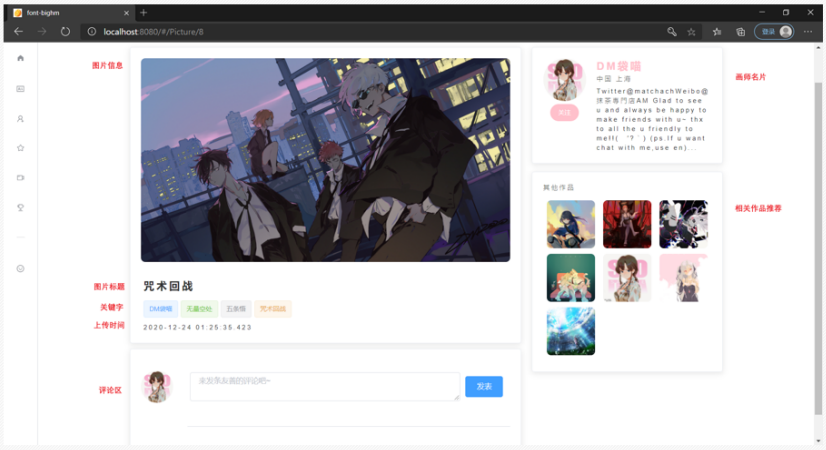


查看与更改个人名片

查看个人名片，包括用户的个人头像、用户名、住址、个性签名等信息

对用户个人姓名、住址、个性签名等信息进行更改

画作详情页



用户可以查看图片的详情信息，比如"图片、标题、关键字、上传时间等

详情页也会显示作品所属画师的个人信息。

同时会向用户给出该画师的相关作品进行推荐。

评论区

用户-评论部分：

1.用户查看画作下面的评论信息

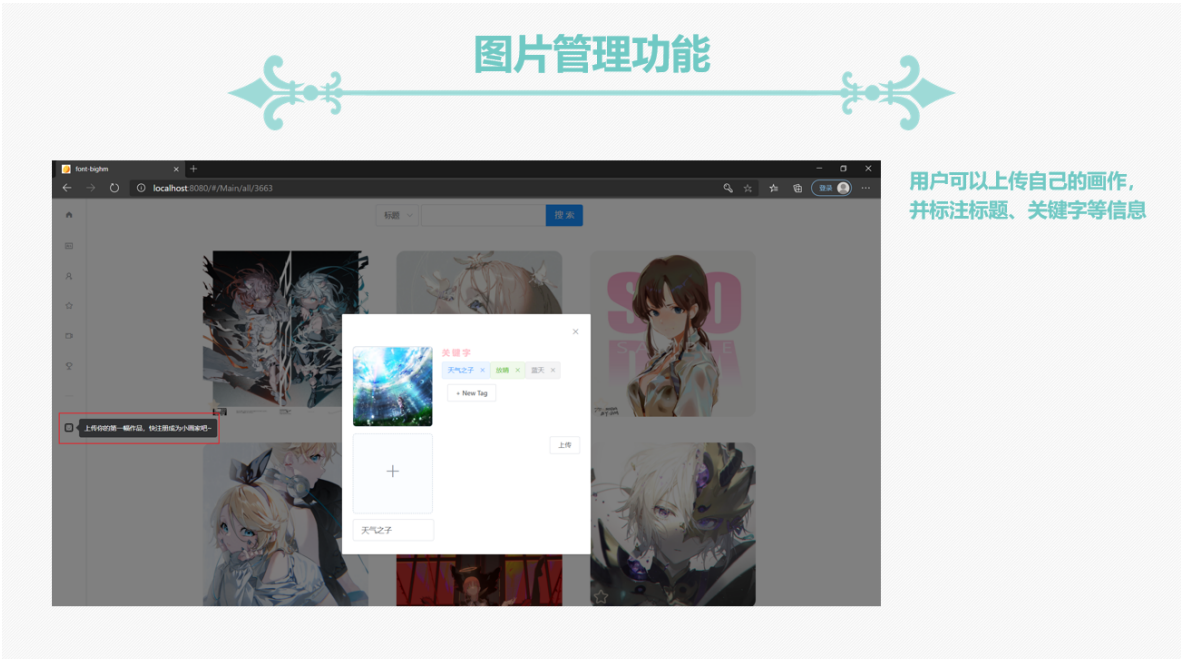
2.用户增、删对某一个画作的评论。

3.用户可以对评论增加和删除点赞信息。

Relua

在设计时，我们希望能保证基础游玩体验和玩法策略的多样化。在钟离的角色设计定位上也是如此。他是一个偏辅助定位的角色，辅助方向以庇护队伍中的其他角色为核心。基于这一点，我们将他最核心的能力，定位在了制造护盾、控制敌人这两点上

2020-12-24 01:25:35.447 1



源程序简要说明

所有基本表的使用都会经过：

基本表定义 --> 基本表与java中类的映射 --> 提供sql访问方法 --> 提供前端访问路径 --> 前端实现视图并通过后端调用数据 --> 用户获得反馈结果

这些过程，故我们以用户基本表为例，进行源程序的说明。

1.基本表定义

```
CREATE TABLE users
(
   userID int identity(1,1) primary key,
   login_name varchar(16) NOT NULL UNIQUE,
   userName varchar(16) DEFAULT '没有名字(⊙__⊙)',
   pass_word varchar(16) NOT NULL,
   sex varchar(4) default '?' check(sex = '男' or sex='女' or sex='?'),
   address varchar(20) default '火星',
   profile_picture varchar(128) DEFAULT 'static/profilePicture/2.jpg',
   signature text NULL default '这个人很懒，什么都没写...'
);
go

```

2.类映射

```
package com.example.bighomework.pojo;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;


@Component
@ConfigurationProperties(prefix = "user")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {
    private int userID;
    private String login_name;
    private String userName;
    private String pass_word;
    private String sex;
    private String address;
    private String profile_picture;
    private String signature; //text -> String?

    public User(String login_name,  String pass_word) {
        this.login_name = login_name;
        this.pass_word = pass_word;
    }

    public User(String login_name, String userName, String pass_word, String sex, String address, String profile_picture, String signature) {
        this.login_name = login_name;
        this.userName = userName;
        this.pass_word = pass_word;
        this.sex = sex;
        this.address = address;
        this.profile_picture = profile_picture;
        this.signature = signature;
    }
}
```

3.提供sql 访问方法

```java
package com.example.bighomework.mapper;

import com.example.bighomework.pojo.User;
import org.apache.ibatis.annotations.*;
import org.springframework.stereotype.Repository;

import java.util.List;

@Mapper
@Repository
public interface UserMapper {

    @Select("select * from LSP.dbo.users")
    public List<User> selectAll();

    @Select("select * from LSP.dbo.users where userID=#{userID}")
    public User selectByID(int userID);

    @Select("select userName from LSP.dbo.users where userName = #{userName}")
    public List<String>  selectIDByName(String userID);

    @Select("select * from LSP.dbo.users where login_name = #{login_name}")
    public User selectUserByLoginName(String login_name);

    @Insert("insert into LSP.dbo.users " +
            "(login_name,  pass_word) " +
            "values " +
            "(#{login_name}, #{pass_word})")
    public void insertUser(User user);

    @Delete("delete from LSP.dbo.users where userID=#{userID}")
    public void deleteUser(int userID);

    @Update("update LSP.dbo.users set " +
            "login_name=#{login_name}, userName=#{userName}, pass_word=#{pass_word} , sex=#{sex}, " +
            "address=#{address}, profile_picture=#{profile_picture}, signature=#{signature}" +
            " where userID=#{userID}")
    public void updateUser(User user);

    @Update("update LSP.dbo.users set userName = #{userName} where userID = #{userID}")
    public void updateUserName(int userID, String userName);

    @Update("update LSP.dbo.users set address = #{address} where userID = #{userID}")
    public void updateUserAddress(int userID, String address);

   @Update("update LSP.dbo.users set signature = #{signature} where userID = #{userID}")
    public void updateUserSignature(int userID, String signature);

```

```
    @Update("update LSP.dbo.users set profile_picture = #{profile_picture} where userID = #{userID}")
    public void updateProfilePicture(int userID, String profile_picture);
}

```

4.提供前端访问路径

```
package com.example.bighomework.controller;

import com.example.bighomework.mapper.PainterMapper;
import com.example.bighomework.mapper.UserMapper;
import com.example.bighomework.pojo.Painter;
import com.example.bighomework.pojo.User;
import jdk.nashorn.api.scripting.JSObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.configurationprocessor.json.JSONObject;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.lang.annotation.Repeatable;
import java.util.List;

@RestController
public class UserController {

    @Autowired
    private UserMapper userMapper;
    @Autowired
    private PainterMapper painterMapper;

    @RequestMapping(value = {"/users/queryAll"}, method = RequestMethod.GET)
    public List<User> queryAllUser() {
        List<User> userList = userMapper.selectAll();
        return userList;
    }


    //begin
    @RequestMapping(value = {"/users/query"}, method = RequestMethod.GET)
    public List<String> queryIDByName(@RequestParam String name) {
        return userMapper.selectIDByName(name);
    }

    @RequestMapping(value = {"/users/query/{id}"}, method = RequestMethod.GET)
    public User queryUserByID(@PathVariable int id) {//can id be int when used in dynamic URL?
        return userMapper.selectByID(id);
    }

    //wait to be tested

    @RequestMapping(value = {"/users/getByID"}, method = RequestMethod.GET)
```

```
    public User getUserByID(@RequestParam int userID) {
        try {
            return userMapper.selectByID(userID);
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("UserController-getUser: 出错啦!");
        }
        return new User();
    }

    @RequestMapping(value = {"/users/get"}, method = RequestMethod.GET)
    public ResponseEntity<String> getUserByLoginName(@RequestParam String login_name, String password) {
        //用户名不存在、密码错误...
        User user = userMapper.selectUserByLoginName(login_name);
        if (user == null) {
            System.out.println(login_name + "用户不存在!");
            return new ResponseEntity<String>("", HttpStatus.INTERNAL_SERVER_ERROR);
        } else if (!password.equals(user.getPass_word())) {
            System.out.println(login_name + "用户密码错误! 错误密码: " + password);
            return new ResponseEntity<String>("", HttpStatus.NOT_IMPLEMENTED);
        } else {
            try {
                JSONObject back = new JSONObject();
                back.put("userID", user.getUserID());
                back.put("login_name", user.getUserName());
                back.put("userName", user.getUserName());
                back.put("pass_word", user.getPass_word());
                back.put("sex", user.getSex());
                back.put("address", user.getAddress());
                back.put("profile_picture", user.getProfile_picture());
                back.put("signature", user.getSignature());

                Painter painter = painterMapper.selectByID(user.getUserID());
                if (painter == null) {
                    back.put("isPainter", false);
                } else {
                    back.put("isPainter", true);
                }

                return new ResponseEntity<String>(back.toString(), HttpStatus.OK);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return new ResponseEntity<String>("", HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
    //end

    @RequestMapping(value = {"users/registe"}, method = RequestMethod.POST)
```

```
95      public ResponseEntity<String> registeUser(@RequestParam String
    login_name, String pass_word) {
96          System.out.println("正在注册用户...");
97          try {
98              //1.先检查用户是否重新注册
99              User repeateUser =
    userMapper.selectUserByLoginName(login_name);
100             if (repeateUser != null) {
101                 //重复登陆
102                 return new ResponseEntity<String>("repeate!",
    HttpStatus.BAD_REQUEST);
103             }
104             //2.否则成功插入
105             User user = new User(login_name, pass_word);
106             userMapper.insertUser(user);
107             return new ResponseEntity<String>("success!", HttpStatus.OK);
108         } catch (Exception e) {
109             e.printStackTrace();
110             return new ResponseEntity<String>("fail!",
    HttpStatus.INTERNAL_SERVER_ERROR);
111         }
112     }
113
114     @RequestMapping(value = {"/delete"}, method = RequestMethod.DELETE)
115     public String deleteUser(@RequestParam int userID) {
116         try {
117             userMapper.deleteUser(userID);
118             return "success!";
119         } catch (Exception e) {
120             e.printStackTrace();
121             return "delete fail!";
122         }
123     }
124
125     @RequestMapping(value = "/users/update", method = RequestMethod.POST)
126     public String upadteUser(@RequestBody String json) {
127         try {
128             JSONObject userparams = new JSONObject(json);
129             int userID = userparams.getInt("userID");//约定: 必须传userID
130             if (userparams.has("userName")) {
131                 userMapper.updateUserName(userID,
    userparams.getString("userName"));
132             } else if (userparams.has("address")) {
133                 userMapper.updateUserAddress(userID,
    userparams.getString("address"));
134             } else if (userparams.has("signature")) {
135                 userMapper.updateUserSignature(userID,
    userparams.getString("signature"));
136             } else if (userparams.has("profile_picture")) {
137                 userMapper.updateUserSignature(userID,
    userparams.getString("profile_picture"));
138             } else {
139                 System.out.println("UserController-updateUser: 错误的参数类
    型");
140             }
141             System.out.println("更新用户信息成功!");
142             return "success!";
143         } catch (Exception e) {
```

```
            e.printStackTrace();
            return "update fail!";
        }
    }
}

```

5.前端视图与调用数据方法

```
<template>
  <el-card :body-style="{ padding: '0px' }">
    <div class="rowclass">
      <div style="display: flex"> <img :src="inputUserProfilePicture" style="align-self: center" class="image ind"></div>

      <div style="margin: 0px;padding: 0px 14px; width: 100%">
        <div style=" width: 100%">
          <div v-if="!updateUserNameStatus" class="individuleProduce username" @click="editName">{{inputUserName}}</div>
          <el-input v-else
                    size="mini"
                    v-model="inputUserName"
                    class="input-class"
                    style=""
                    @keydown.enter.native="updateUserName">
          </el-input>
          <!--          <i class="el-icon-edit" style="color: pink; float: right" @click="updateUserMsg"></i>-->
        </div>
        <div v-if="!updateUserAddressStatus" @click="editAddress" class="individuleProduce useraddress" >{{inputUserAddress}}</div>
        <el-input v-else
                  size="mini"
                  v-model="inputUserAddress"
                  class="input-address-class"
                  @keydown.enter.native="updateUserAddress">
        </el-input>
        <div v-if="!updateUserTextStatus" @click="editText" class="bottom clearfix individuleProduce usertext">
          {{inputUserSignature}}
        </div>
        <el-input v-else
                  size="mini"
                  type="textarea"
                  v-model="inputUserSignature"
                  style="width: 100%; margin-top: 10px; margin-bottom: 7px; margin-left: 5px"
                  @keydown.enter.native="updateUserSignature">
        </el-input>
      </div>
    </div>
    <my-user-update-message ref="updateMessage"/>
  </el-card>
</template>

<script>
  import myUserUpdateMessage from '../../../components/User/UpdateMessage'
```

```
43
44  export default {
45    name: "IndividualCard",
46    components: {
47      myUserUpdateMessage
48    },
49
50    data() {
51      return {
52        inputUserName: "",
53        inputUserAddress: "",
54        inputUserSignature: "",
55        inputUserProfilePicture: "",
56
57        updateUserNameStatus: false,
58        updateUserAddressStatus: false,
59        updateUserTextStatus: false,
60        updateUserProfilePictureStatus: false,
61      }
62    },
63
64    methods: {
65      editName() {
66        this.updateUserNameStatus = true;
67      },
68      editAddress() {
69        this.updateUserAddressStatus = true;
70      },
71      editText() {
72        this.updateUserTextStatus = true;
73      },
74      editPicture() {
75        this.updateUserProfilePictureStatus = true;
76      },
77
78      debug() {
79        console.log(this.userID, this.userName, this.address);
80      },
81
82      updateUserName() {
83        this.updateUserNameStatus = false;
84        this.$refs.updateMessage.updateUserName(this.inputUserName);
85      },
86
87      updateUserAddress() {
88        this.updateUserAddressStatus = false;
89        this.$refs.updateMessage.updateUserAddress(this.inputUserAddress);
90      },
91
92      updateUserSignature() {
93        this.updateUserTextStatus = false;
94
  this.$refs.updateMessage.updateUserSignature(this.inputUserSignature);
95      },
96
97      updateUserProfilePicture() {
98        this.updateUserProfilePictureStatus = false;
```

```
 this.$refs.updateMessage.updateUserProfilePicture(this.inputUserProfilePicture);
      },

      updateUserMsg() {
        console.log("正在更改用户信息...");
        this.$refs.updateMessage.open();
      }
    },

    created() {
      //1.拉取用户信息
      this.inputUserName = this.$store.state.user.userName;
      this.inputUserAddress = this.$store.state.user.address;
      this.inputUserSignature = this.$store.state.user.signature;
      this.inputUserProfilePicture = this.$store.state.user.profile_picture;
    }
  }
</script>
```

其它基本表的使用大致上等同于上述方法，因此不再赘述。

收获与体会

数据库方面，学习了关系数据库、关系操作以及关系的完整性。了解了sql基本概念，如何对数据进行定义查询与更新。学习了如何对数据库进行一些基本的安全性保护。学习了对数据库进行较为规范化的设计，如需求分析、概念结构设计、逻辑结构设计等。

后端方面，学会了如何进行数据库与后端的连接以及用户与数据库的交互。了解了spring-boot与AOP与IOC设计模式。前端方面，学习了html,css,javascript等语言，vue框架，页面设计与视图管理。