登录注册功能实现

201870202 任俊宇

实现内容

1. 通过node.js的express框架实现服务器注册和登录功能，详见app.js

```
app.post( path: '/api/register', handlers: async(req : ... , res : Response<ResBody, Locals> )=>{
    var username = String(req.body.username);
    var password = String(req.body.password);
    console.log(username, password);
    const user1 = await User.findOne({
        username: req.body.username
    })
    if (user1){
        return res.status( code: 422).send( body: {message: 'Username already exists!'})
    }
    const user = await User.create({
        username,
        password
    })
    res.send(user);
})
app.post( path: '/api/login', handlers: async(req : ... , res : Response<ResBody, Locals> )=>{
    const user = await User.findOne({
        username: req.body.username,
    })
    if (!user){
        return res.status( code: 422).send( body: {message: 'Username does not exist!'})
    }
    const isPasswordValid = require('bcrypt').compareSync(req.body.password, user.password);
    if (!isPasswordValid){
        return res.status( code: 422).send( body: {message: 'Password is incorrect'});
    }
    res.send(user);
})
```

2. 通过Mongodb数据库存放用户的数据，详见models.js
3. 通过bcrypt方案实现了密码加密，详见models.js

```
const mongoose = require('mongoose');
mongoose.connection.once('open',()=>{
    console.log('数据库连接成功')
});
mongoose.connect('mongodb://localhost/user',{
    useNewUrlParser:true,
});
const User = mongoose.model( name: 'User',new mongoose.Schema( definition: {
    username:{type: String, unique: true},
    password:{type: String, set(val){
        return require('bcrypt').hashSync(val, salt: 10);
        }}
}));
module.exports = {User};
```

4. 通过Ajax发送post请求，详见login.html和signup.html

```
$.ajax({
    type:"POST",
    dataType:"json",
    crossDomain:true,
    url:'http://localhost:3000/api/login',
    contentType:"application/json",
    data:JSON.stringify({
        'username' : username,
        'password' : password,
    }),
    success:function (result){
        console.log(result);
        alert("Sign in succeed!");
        window.location.href="main.html";
    },
    error:function (result){
        console.log(result);
        alert(result.responseJSON.message);
    }
})
```

5. 实现了了验证码验证，详见 login.html，通过 createCode()函数生成随机的验证码，在发送请求前验证验证码正确与否，若验证码正确才发送 post 请求

```
function createCode() {
    code = [];
    var len = 4;
    var vcode = document.getElementById("vcode");
    vcode.value = "";
    var char = [2, 3, 4, 5, 6, 7, 8, 9,'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J',
    for (var i=0;i<len;+i++){
        var index = Math.floor(Math.random()*32);
        code += char[index];
    }
    if (code.length !== len){
        createCode();
    }
    vcode.value = code;
}
```

6. 采用正则表达式规定了密码的格式，规定了用户名为4-20味道字母或数字，密码为6-16为的字母或数字

```
<span>Username</span>
<input class="input" id="username" type="text" placeholder="Please enter your username..." pattern="^[a-zA-Z0-9]{3,19}$" name="Username">
<br>
<span>Password</span>
<input class="input" id="password" type="password" placeholder="Please enter your password..." pattern="^[a-zA-Z0-9]{5,15}$" name="Password">
```

7. 实现了简单的密码强度提示，当密码为空时提示"password is blank"；当密码不足 6 位时提示"not long enough"；当密码为 6-10 位时提示"could be stronger"；当密码为 10-16 位时提示"strong enough"；当密码超过 16 位时提示"too long"。详见signup.html

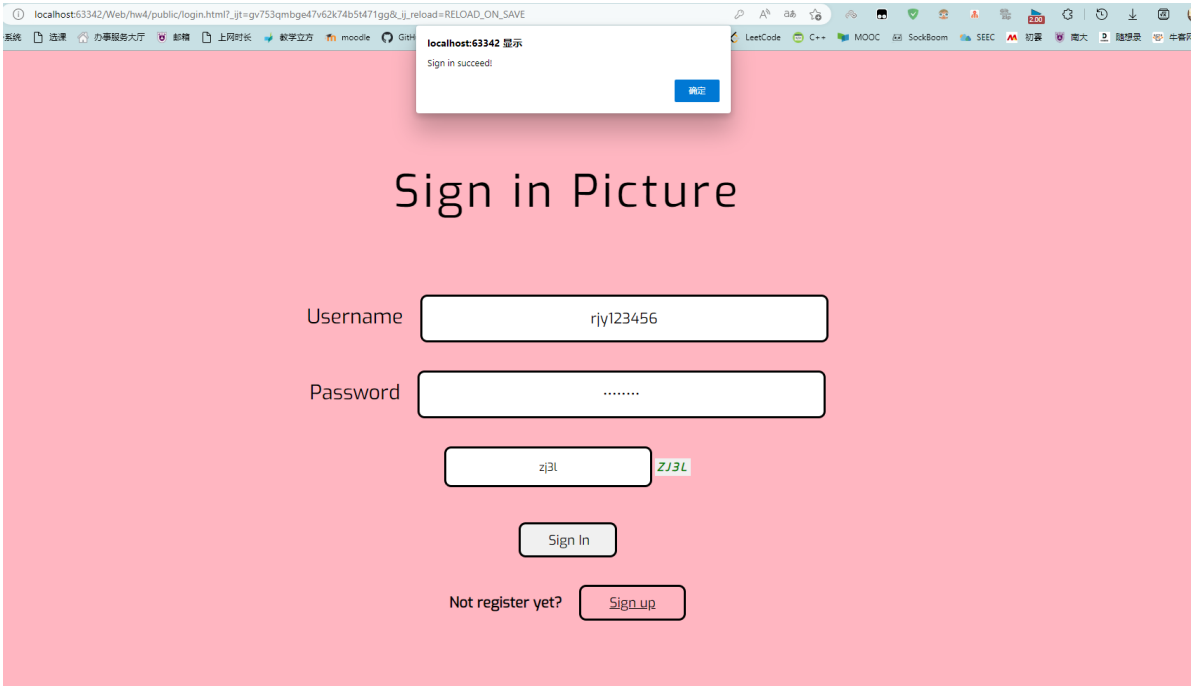
```
$(document).ready(function (){
    5+ 个用法
    const changeText = function (el, text, color){
        el.text(text).css('color', color);
    };

    $('.input1').keyup(function (){
        let len = this.value.length;
        const pbText = $('.progress-bar_text');
        if (len === 0){
            changeText(pbText, 'Password is blank');
        }
        else if (len > 0 && len < 6){
            changeText(pbText, 'Not long enough');
        }
        else if (len >= 6 && len < 10){
            changeText(pbText, 'Could be stronger');
        }
        else if(len>=10 && len <=16){
            changeText(pbText, 'Strong password');
        }
        else{
            changeText(pbText, 'Too long');
        }
    });
});
```

实现界面

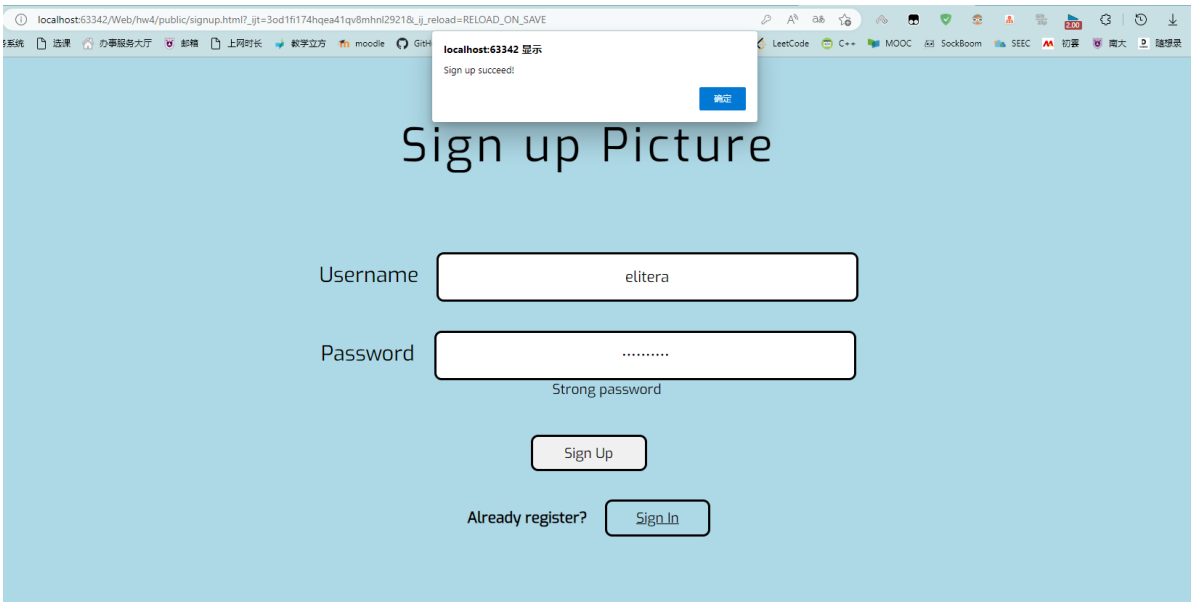
服务器启动

```
PS C:\Code\WebStorm\Web\hw4> node app.js
(node:3920) [MONGOOSE] DeprecationWarning: Mongoose: the `strictQuery` option will be switched back to `fa
want to prepare for this change. Or use `mongoose.set('strictQuery', true);` to suppress this warning.
(Use `node --trace-deprecation ...` to show where the warning was created)
http://localhost:3000
数据库连接成功
```

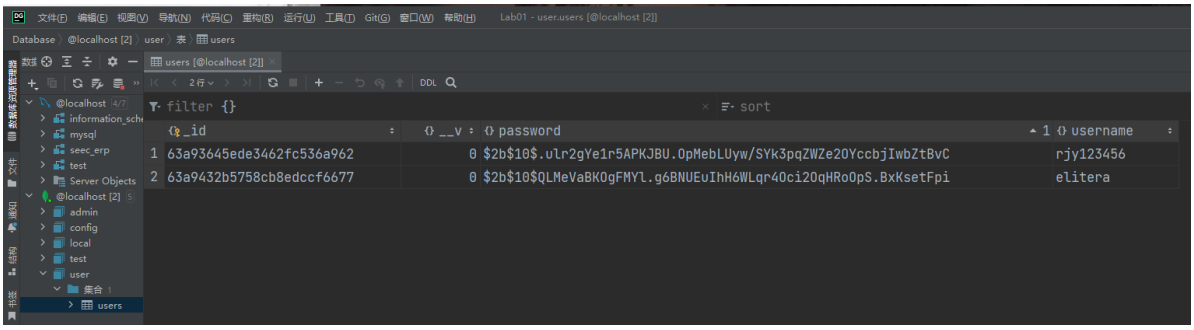
登陆成功


localhost:63342 显示
Sign in succeed!
确定
Sign in Picture
Username
rjy123456
Password
zj3l
ZJ3L
Sign In
Not register yet?
Sign up

注册成功


localhost:63342 显示
Sign up succeed!
确定
Sign up Picture
Username
elitera
Password
Strong password
Sign Up
Already register?
Sign In

数据表


Database
@localhost [2]
user
表
users
filter {}
sort
_id
__v
password
username
1 63a93645ede3462fc536a962 0 $2b$10$.ulr2gYe1r5APKJBU.OpMebLUyw/SYk3pqZWZe2OYccbjIwbZtBvC rjy123456
2 63a9432b5758cb8edccf6677 0 $2b$10$QLMeVaBKOgFMYl.g6BNUEuIhH6WLqr4Oci2OqHRoOpS.BxKsetFpi elitera