华容道项目的技术难点

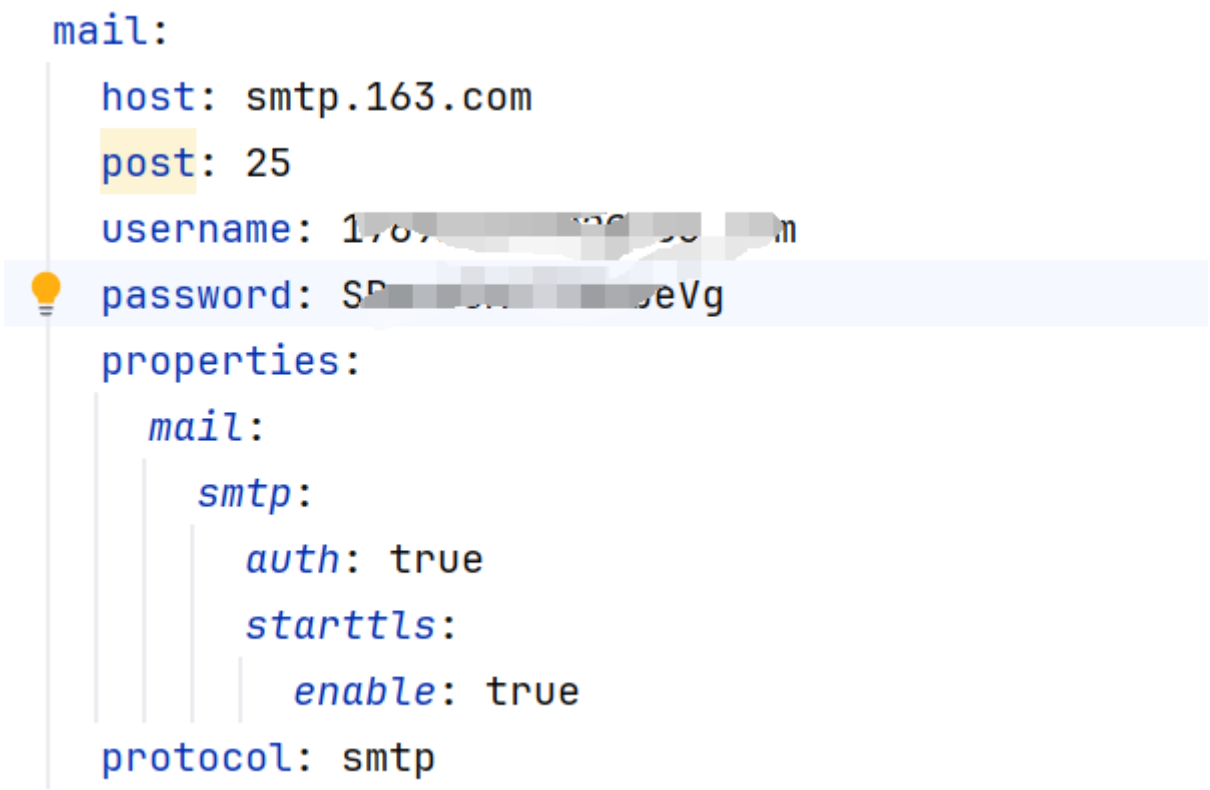
1. 注册功能如何实现

1.开启自己的一个邮箱的SMIP服务，之后就能通过代码来控制邮箱发送邮件。

2.确保xml文件中引入了mail依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

3.配置yml文件



这里介绍一下这个配置什么意思：

最下面的protocol是使用的协议，上面开的是STMP，就使用smtp协议

然后这个host是这个邮箱对应的这个协议的服务器的url，如下图，网易邮箱的SMTP服务器地址为 smtp.163.com

然后是port，这个是通信使用的具体端口号，可以去官方文档查看，或者去问ai “xx邮箱的xxxx协议的默认端口号是多么”。这里网易邮箱smtp协议的端口号是25

username是你邮箱的用户名 比如12411015@mail.sustech.edu.cn

password是开启SMTP协议时会让你设置一个授权码，这个位置就填写你的授权码。

最后是下图这个部分：

```yaml
properties:
  mail:
    smtp:
      auth: true
      starttls:
        enable: true
```

properties是用于定义邮件相关属性的配置块

其中smtp：auto：true是开启SMTP验证，这一步必须做，开启后会根据你填写的username和password进行验证，如果你的用户名和密码不对，就不会自动发送邮件。

下方的smtp:starttls:enable:true是启用 STARTTLS 扩展 ，它可以将明文通信升级为加密通信，增强邮件传输过程中的安全性，这个不是必要的。类似的不是必要，但是可以提高安全性，类似的有趣的

配置还有

加密相关

- `mail.smtp.socketFactory.port`： 指定用于 SSL/TLS 加密连接的端口。比如，若使用 SSL 加密连接到 SMTP 服务器，可设置此端口为 465 。
- `mail.smtp.socketFactory.class`： 指定用于创建 SSL/TLS 套接字的类。常见值如 `javax.net.ssl.SSLSocketFactory` ，用于启用 SSL 加密连接 。
- `mail.smtp.ssl.enable`： 明确开启 SSL 加密，设置为`true` 时强制使用 SSL 进行 SMTP 连接。

调试相关

- `mail.debug`： 设置为`true` 时，会在控制台或日志中打印邮件发送过程中的详细调试信息，便于排查配置或发送过程中的问题 。

连接属性相关

- `mail.smtp.connectiontimeout`： 设置连接到 SMTP 服务器的超时时间（单位为毫秒 ），如设置为`5000` 表示连接尝试超 5 秒则失败 。
- `mail.smtp.timeout`： 设置与 SMTP 服务器通信时数据传输的超时时间（单位毫秒 ），防止长时间等待无响应的连接。
- `mail.smtp.writetimeout`： 设置向 SMTP 服务器写入数据时的超时时间（单位毫秒 ）。

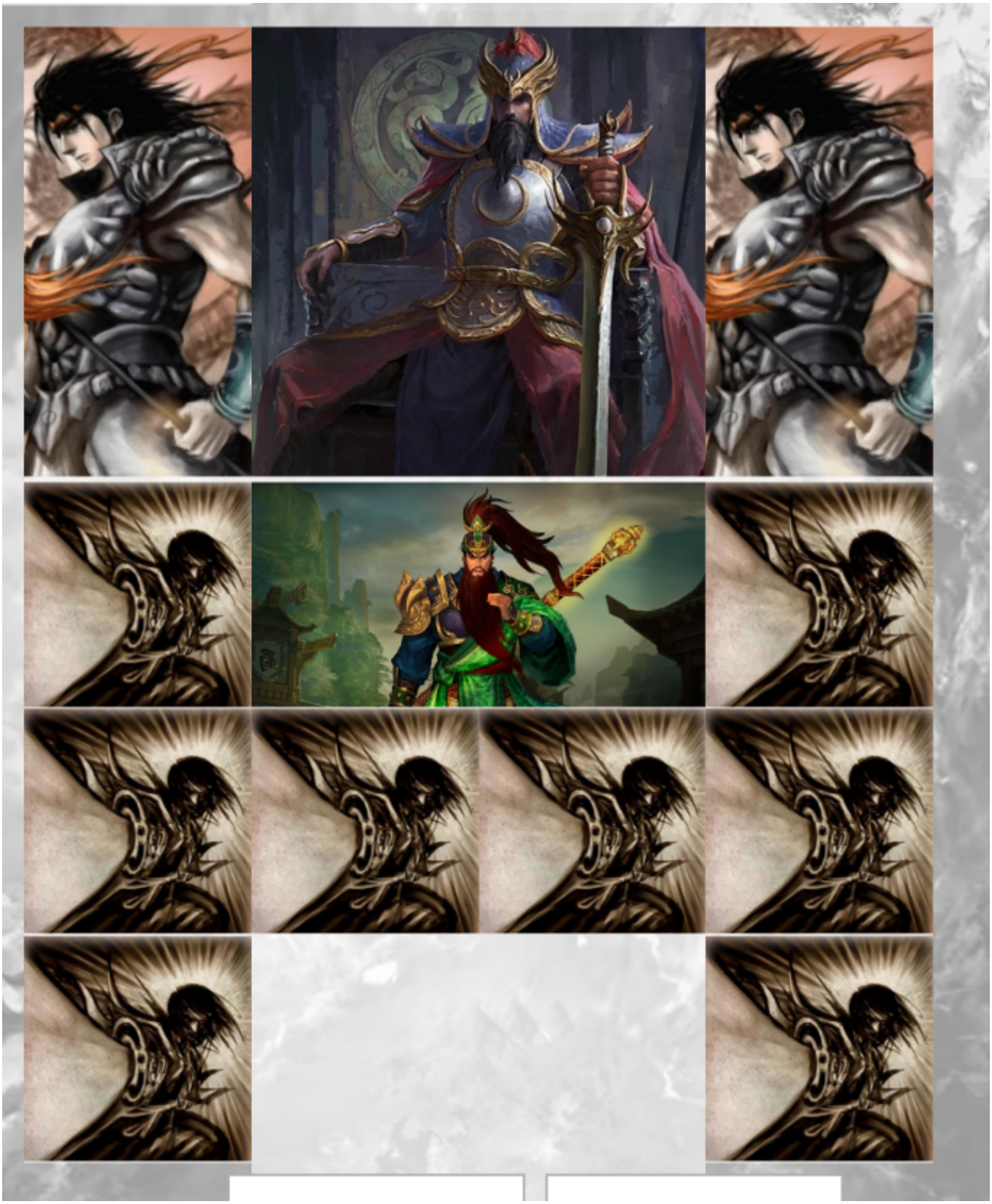
4.开始编写REgistrationController类，就两个功能，1.发送验证码，2.点击注册按钮时将用户注册入数据库

`RegistrationController` 是一个用于处理用户注册相关功能的 Spring Boot 控制器类。它包含两个主要的端点：

1. **发送验证码 (`sendVerificationCode`)**：
    - 使用 `GET` 请求，接收用户的邮箱地址。
    - 验证邮箱地址是否有效。
    - 生成一个随机的六位验证码，并将验证码及其过期时间存储在当前会话的 `HttpSession` 中。
    - 调用 `RegistrationService` 的 `sendEmail` 方法发送验证码到用户邮箱。
    - 返回操作结果。
2. **注册用户 (`register`)**：
    - 使用 `POST` 请求，接收用户的注册信息（邮箱、用户名、密码、确认密码、验证码）。
    - 验证验证码是否正确及是否过期。
    - 检查两次输入的密码是否一致。
    - 创建一个 `User` 对象并设置默认属性（如头像 URL 和初始奖励数）。
    - 调用 `RegistrationService` 的 `add` 方法将用户信息存储到数据库。
    - 返回操作结果。

然后写一个前端，两个按钮。一个“发送验证码”按钮，点击后就调用方法1.发送验证码。一个“注册”按钮，点击后就调用方法2.注册用户。

2. 华容道棋盘的实现



华容道是一个5x4的棋盘，其中符合

赵云（1）	曹操（1）	曹操（2）	赵云（1）
赵云（2）	曹操（3）	曹操（4）	赵云（2）
兵	关羽（1）	关羽（2）	兵

兵	兵	兵	兵
兵	空白	空白	兵

我们可以规定

* 0 空位

* 1 实体（我们只规定一个大于1x1的块的左上角代表该块，其余部分用“1实体”代替。比如上图曹操，除了曹操（1）外的2、3、4均用实体代替）

* 2 单兵

* 3 竖行（赵云）

* 4 横行（关羽）

* 5 曹操

赵云（1）	曹操（1）	实体	赵云（1）
实体	实体	实体	实体
兵	关羽（1）	实体	兵
兵	兵	兵	兵
兵	空白	空白	兵

然后可以用数字代替其中块

3	5	1	3
1	1	1	1
2	4	1	2
2	2	2	2
2	0	0	2

转换为字符串“35131111241222222002”。然后我们就以一个字符串代替一个**布局情况**。改变布局情况就是改变字符串，然后重新渲染。比如我要把位于纵4横2位置的兵向下移动一个

3	5	1	3
1	1	1	1
2	4	1	2
2	0	2	2
2	2	0	2

就是把字符串改为“35131111241220222202”

然后就是注册点击事件。就是Component类中Grid.vue中的方法，其实最开始想做一个也可以在移动端游玩的功能，所以有这三个方法。

```
handleTouchStart (e) {
    this.previousX = e.targetTouches[0].clientX;
    this.previousY = e.targetTouches[0].clientY;
    this.draging = true;
},
handleTouchMove (e) {
    let currentX = e.targetTouches[0].clientX;
    let currentY = e.targetTouches[0].clientY;


    this.x += currentX - this.previousX;
    this.y += currentY - this.previousY;

    this.previousX = currentX;
    this.previousY = currentY;

    let shiftX = this.x - this.startX;
    let shiftY = this.y - this.startY;
    let direction = 0;
    if (Math.abs(shiftY) > Math.abs(shiftX)) {
        if (-shiftY > this.unitSize / 1.5) direction = 1;
        else if (shiftY > this.unitSize / 1.5) direction = 3;
    } else {
        if (shiftX > this.unitSize / 1.5) direction = 2;
        else if (-shiftX > this.unitSize / 1.5) direction = 4;
    }
    this.handleMove(direction, this.position);
},
handleTouchEnd (e) {
    let _this = this, step = 5;
    let distanceX = _this.x - this.startX;
    let distanceY = _this.y - this.startY;
    let inv = setInterval(() => {
        if (!--step) { clearInterval(inv); _this.draging = false; }
        _this.x = this.startX + distanceX * step * 0.1;
        _this.y = this.startY + distanceY * step * 0.1;
    }, 10);
},
```

但是最后只实现了鼠标端功能，就是在触碰移动下方的三个关于鼠标移动的方法。

然后键盘移动方法在Move.vue类中。

移动的核心是判断这次移动是否有效，具体判断是否有效在src/api/core.js中。

3. AI算法的实现

理论上实现这类棋类AI需要使用蒙特卡洛算法，但是本人不是很会，所以使用了别的方法。

其实华容道每一个布局都有当前的解。

如下图，其实胜利条件就是5到底纵4横2的位置。**就是字符串“35131111241222222002”通过符合规则的变换后，能让5到达下标为13的位置**

3	5	1	3
1	1	1	1
2	4	1	2
2	2	2	2
2	0	0	2

使用**广度优先算法**来寻找当前布局的解。

在core.js类中有该算法的实现。注意：**该算法是可以算出解，但不一定是唯一解！**

```
let getSolve = function (state) {
    let que = [state], vst = { [state]: 1 }, result = [];

    while (que.length) {
        let cur = que.shift(), res = false;

        if (cur[13] === '5') {
            for (; cur !== 1; cur = vst[cur])
                result.push(cur);
            result.pop();
            break;
        }

        for (let i = 0; i < cur.length; i++) {
            (res = moveUp(cur, i)) && !vst[res] && que.push(res) && (vst[res] = cur);
            (res = moveDown(cur, i)) && !vst[res] && que.push(res) && (vst[res] = cur);
            (res = moveLeft(cur, i)) && !vst[res] && que.push(res) && (vst[res] = cur);
            (res = moveRight(cur, i)) && !vst[res] && que.push(res) && (vst[res] = cur);
        }
    }
    return result;
}
```

简单来讲就是创建一个Map名叫vst，用于存储“移动后的布局”：“当前布局”

再创建一个队列，其中的元素都是“当前布局”，然后while循环中弹出队列最前方的判读是否是最终胜利状态（下标为13的位置为“5”）。

如果不是，就广度搜索棋盘中的每个位置的四个方向的移动。

大部分的移动是无效的，会直接返回false，进而阻止&&之后的部分的运行。

如果移动是有效的，判断这个移动后的布局是否在vst中（这个是为了防止一个块向上移动，然后向下移动，然后再向上移动，这样反复的情况，同时可能导致忽略掉一些可能的解。），如果没有，就将其放入队列，并存储“移动后的布局”：“当前布局”。

然后循环判断队列中的下一个“当前布局”，如法炮制，直到出现下标为13的位置为“5”的“当前布局”，证明胜利了。

然后读取vst映射集合：

“胜利布局”：“胜利布局的上一个布局”，然后将胜利布局放入result栈中

“胜利布局的上一个布局”：“胜利布局的上一个布局的上一个布局”，然后将胜利布局的上一个布局放入result数组中

“胜利布局的上一个布局的上一个布局”：“胜利布局的上一个布局的上一个布局的上一个布局”，然后将胜利布局的上一个布局的上一个布局放入result数组中

以此类推，得到胜利布局到当前布局的栈，然后逐个吐出，然后根据字符串渲染棋局，就能形成AI解答的动画。提示下一步就是只吐出一步的AI动画

然后每次点击“提示下一步”或者“AI解答”都会重新进行用这个算法计算。

4. 悔棋

一个栈存储之前的字符串，点击“悔棋”后就使用弹出上一步的字符串渲染棋局

5. 存档

使用数据库存储当前用户的用户名和当前棋局字符串。

读档时使用这个字符串渲染棋局

6. 观战模式

数据库中建一个表，

当用户进入游戏模式的url时，表中创建一个对象，存储当前用户的用户名，棋局字符串。同时创建一个url，以该用户名为结尾。

当用户改变棋局时，改变存储的该用户名的棋局字符串。

当用户退出游戏模式时，销毁该用户名的对象。

当有用户点击观战后，前端初始化websocket请求。后端接收情况，建立websocket对话。

每当检测到被观战的用户的前端的layout发生变化，就改变数据库中layout，然后通过前端广播给所有建立了websocket连接的观战用户。

当观战用户退出，断开websocket连接。