前端代码风格

组件名为多个单词，避免与 HTML 元素冲突

```
<div class="loginBox">
  <div style="width: 100%; height: 40%; text-align: center; overflow: hidden">
```

使用了规范的缩进

```
    <script setup>
    import { Avatar, ElementPlus, Lock } from "@element-plus/icons-vue";
    import { ref } from "vue";

    defineEmits(["no-account-register"]);
    </script>

    <template>
      <div class="loginBox">
        <div style="width: 100%; height: 40%; text-align: center; overflow: hidden">
          <img src="../assets/ERoad-logo.png" style="width: 100%; height: 100%" />
        </div>
        <div
          style="
            display: flex;
            align-items: center;
            justify-content: center;
            width: 100%;
            height: 60%;
            flex-grow: 0;
          "
        >
          <el-form :model="user" style="width: 80%" :rules="rules" ref="loginRef">
            <div
              style="
                font-size: 20px;
                font-weight: bold;
                text-align: center;
                margin-bottom: 20px;
              "
            >
```

合理使用了注释

```
    //bound the region
        var options = {
            areas: [{
              // visible: false, // 是否可见
              rejectTexture: true, // 是否屏蔽自定义地图的纹理
              path: [[
              [116.303959,39.997705],
              [116.309087,40.000302],
              [116.315761,40.000319],
              [116.317306,39.985852],
              [116.304946,39.985524]
              ]]
            }]
          }
          // 外多边形坐标数组和内多边形坐标数组
          var outer = [
            new AMap.LngLat(-360, 90, true),
            new AMap.LngLat(-360, -90, true),
            new AMap.LngLat(360, -90, true),
            new AMap.LngLat(360, 90, true),
          ]
          var pathArray = [ outer ]
          pathArray.push.apply(pathArray, options.areas[0].path) // options.areas[0].path 外部区域 遮罩
          // pathArray = [ outer ] // 整个地图遮罩
          // pathArray = options.areas[0].path // options.areas[0].path 内部区域 遮罩
```

合理使用了换行

```
</script>

<style scoped>
#container {
  position:fixed;
  top:0;
  left:0;
  background-size:cover;
  background-repeat:no-repeat;
  background-position: center center;
  width: 100%;
  height: 100%;
}
#controller{
  position: absolute;
  z-index: 99;
  top: 20px;
  left: 100px;
  background: rgb(255, 255, 255);
  list-style-type: none;
  width: 120px;
}
</style>
```

一个模块写完后用空行进行分割

```
} from "@element-plus/icons-vue";
import { ref } from "vue";
defineEmits(["have-account-login"]);
</script>

<template>
  <div class="registerBox">
    <div style="width: 100%; height: 40%; text-align: center; overflow: hidden">
      <img src="../assets/ERoad-logo.png" style="width: 100%; height: 100%" />
    </div>
```

后端代码风格

使用了良好的缩进规范

```
for code in verifiycode:
    if (create_time - code.create_time).seconds < 60:
        return Response(data={
            "msg": "please wait for 60 seconds",
            "status": status.HTTP_400_BAD_REQUEST
        })
    if (create_time - code.create_time).seconds > 600:
        code.delete()
```

进行了合理的注释

```
47      def validate_username(self, username):
48          # 检查长度
49          # password = super.validate_password(password)
50          if len(username) < 4 or len(username) > 20:
51              raise serializers.ValidationError(
52                  "Username Length Error",
53              )
54          # 检查是否由字母与数字组成
55          if not username.isalnum():
56              raise serializers.ValidationError("Username Format Error")
57          # 检查是否有重复用户名
58          # if AppUser.objects.filter(username=username).exists():
59          #     raise serializers.ValidationError("Username already exists")
60          return username
```

单行代码长度在控制在合理范围内

```
88      def register(self, request):
89          username = request.data.get('username')
90          password = request.data.get('password')
91          email = request.data.get('email')
92          confirm = request.data.get('confirm')
93          verify_code = request.data.get('verify_code')
94          if confirm != password:
95              return Response(data={
96                  "msg": "Password not match",
97                  "status": status.HTTP_400_BAD_REQUEST
98              })
99
100         # 验证验证码是否正确
101         verified = False
102         time_now = timezone.now()
103         verifiycode = VerifyCode.objects.filter(email=email)
104         for code in verifiycode:
105             if code.code == verify_code and (time_now - code.create_time).seconds < 600:
106                 verified = True
107                 break
108             if (time_now - code.create_time).seconds > 600:
109                 code.delete()
110         if not verified:
111             return Response(data={
112                 "msg": "Invalid verify code",
113                 "status": status.HTTP_400_BAD_REQUEST
114             })
```

函数名使用了两个单词，变量名使用了一个单词，且能准确表示变量指代的含义

```
def send_email(self, request):
    email = request.data.get('email')
    create_time = timezone.now()
```

代码不同模块之间合理空行

```
62  TEMPLATES = [
63      {
64          'BACKEND': 'django.template.backends.django.DjangoTemplates',
65          'DIRS': [],
66          'APP_DIRS': True,
67          'OPTIONS': {
68              'context_processors': [
69                  'django.template.context_processors.debug',
70                  'django.template.context_processors.request',
71                  'django.contrib.auth.context_processors.auth',
72                  'django.contrib.messages.context_processors.messages',
73              ],
74          },
75      },
76  ]
77
78  WSGI_APPLICATION = 'backend.wsgi.application'
79
80
81  # Database
82  # https://docs.djangoproject.com/en/4.1/ref/settings/#databases
83
84  DATABASES = {
85      'default': {
86          'ENGINE': 'django.db.backends.sqlite3',
87          'NAME': BASE_DIR / 'db.sqlite3',
88      }
89  }
```