基于神经网络的设备寿命预测 Web 应用及系统构建方法

- 源代码文档

版本	3.0
文档状态:	编辑
作者:	杨烜赫
负责人:	杨烜赫
创建日期:	2024年8月15日
更新日期:	2025年1月15日

修订历史

日期	版本	修改者	描述
2024-8-15	1.0	杨烜赫	进行初稿的撰写
2024-10-11	2.0	杨烜赫	进行进一步的修改与完善
2025-1-15	3.0	杨烜赫	进行进一步的修改与完善

前端部分代码

```
<script setup lang="ts">
import {ref} from "vue";
import {useRouter} from "vue-router";
import {ElFormItem, ElForm, ElCard, ElButton, ElInput, ElMessage} from "element-plus";
import axiosInstance from "../../axios";
import type {FormInstance} from 'element-plus'
import {useAuth} from "../../composables/auth.ts"

const router = useRouter()
const auth = useAuth()
const formRef = ref<FormInstance | null>(null)
const fromRules = ref({
  username: [
    {required: true, message: '请输入用户名', trigger: 'blur'}
  ],
  password: [
    {required: true, message: '请输入密码', trigger: 'blur'}
  ]
})
const form = ref({
  username: '',
  password: '',
})

const login = async () => {
  if (!formRef.value) {
    ElMessage.error('请输入用户名和密码')
    return
  }
  await formRef.value?.validate((valid, _) => {
    if (valid) {
      const url = '/user/login'
      const data = {
        username: form.value.username,
        password: form.value.password
      }
      const headers = {
        'Content-Type': 'application/x-www-form-urlencoded'
      }
      axiosInstance.post(url, data, {headers}).then((res) => {
        if (res.data.code === 200) {
          ElMessage.success('登录成功')
          auth.setToken(res.data.data.token)
          router.push({
            path: "/home",
            replace: true
          });
        } else {
          ElMessage.error('登录失败，请检查用户名和密码')
          console.log(res.data)
        }
      })
    } else {
      ElMessage.error('登录失败，请检查用户名和密码')
    }
  })
}
</script>
<template>
  <div>
    <el-card class="box-card" style="width: 400px;margin: 100px auto;">
      <h2>登录</h2>
      <el-form ref="formRef" :model="form" :rules="fromRules" label-width="80px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="form.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
          <el-input v-model="form.password" type="password"></el-input>
        </el-form-item>
        <div class="btnGroup">
          <el-button type="primary" @click="login">登录</el-button>
          <router-link to="/register">
            <el-button style="margin-left:10px">注册</el-button>
          </router-link>
        </div>
      </el-form>
    </el-card>
  </div>
</template>
<script setup lang="ts">
import { ElMenu, ElSubMenu, ElIcon, ElMenuItem, ElText } from 'element-plus'
import { ref } from 'vue'
import { useRouter } from 'vue-router';

const router = useRouter()
const MenuList = ref([
    {
        index: '1',
        title: '首页看板',
        icon: 'HomeFilled',
        children: [
            {
                index: '1-1',
                title: '首页看板',
            },
        ]
    },
    {
        index: '2',
        title: '模型中心',
        icon: 'Menu',
        children: [
            {
                index: '2-1',
                title: '模型中心',
            },
        ]
    },
    {
        index: '3',
        title: '数据中心',
        icon: 'Histogram',
        children: [
            {
                index: '3-1',
            title: '数据中心',
         }
      ]
   },
   {
      index: '4',
      title: '组件中心',
      icon: 'HelpFilled',
      children: [
         {
            index: '4-1',
            title: '组件列表',
         },
         {
            index: '4-2',
            title: '添加组件',
         }
      ]
   }
])
//@ts-ignore
const handleSelect = (index, indexPath) => {
   // 上级 html 路由为：<el-main><router-view name = "home-main" /></el-main>，我们只对 home-main 部
分进行路由跳转
   console.log(index, indexPath)
   switch (index) {
      case '1-1':
         router.push({ name: 'home-overview'});
         break;
      case '2-1':
         router.push({ name: 'model-recorder' });
         break;
      case '3-1':
         router.push({ name: 'data-recorder' });
         break;
      case '3-2':
         router.push({ name: 'target-notification' });
         break;
      case '4-1':
         router.push({ name: 'analyse-trans' });
         break;
      case '4-2':
         router.push({ name: 'component-add' });
         break;
      default:
         break;
   }
}

</script>

<template>
   <el-menu default-active="1" @select="handleSelect">
      <el-sub-menu v-for="(item, i) in MenuList" :key="i" :index="item.title" :icon="item.icon">
         <template #title>
            <el-icon size="32">
               <!--这里插入的标签名称对应于 item.icon-->
               <component :is="item.icon"></component>
            </el-icon>
            <h3>{{ item.title }}</h3><!--父菜单名称-->
            </template>
            <el-menu-item v-for="(child, j) in item.children" :key="j" :index="child.index">
                <el-text size="large">
                    <h4>{{ child.title }}</h4>
                </el-text>
            </el-menu-item>
        </el-sub-menu>
    </el-menu>
</template>

<script setup lang="ts">
import {
  ElHeader,
  ElMain,
  ElContainer,
  ElAside,
  ElAvatar,
  ElDropdown,
  ElDropdownMenu,
  ElDropdownItem,
  ElRow,
  ElAffix,
  ElCol
} from 'element-plus';
import Menu from '../menu/Menu.vue'
import { useAuth } from '../../composables/auth.ts'
import { useRouter } from 'vue-router'
import axiosInstance from "../../axios";
import {ref,onMounted} from "vue";
const auth = useAuth()
const router = useRouter()
const avatar = ref()
const defaultUrl='https://cube.elemecdn.com/0/88/03b0d39583f48206768a7534e55bcpng.png'
const avatarUrl = ref('')
const logout = () => {
  auth.logout()
  router.push({
    path: "/login",
    replace: true
  });
}
const getUserAvatar = async () => {
  const url = '/user/avatar'
  axiosInstance.get(url,{ responseType: 'blob' }).then((res) => {
    if (res.status === 200) {
      avatar.value = res.data
      const render = new FileReader();
      render.onload = (e) => {
        avatarUrl.value = e.target?.result as string
      }
      render.readAsDataURL(avatar.value)
    } else {
      avatarUrl.value=defaultUrl
    }
  })
}

onMounted(() => {
  avatarUrl.value = defaultUrl
  getUserAvatar()
})
</script>
<template>
  <div class="common-layout" style="position: absolute;top: 0;left: 0;right: 0;bottom: 0;">
    <el-container style="height: 100vh;">
      <el-header height="">
        <el-affix :offset="0">
          <div class="system-header" style="text-align: left;">
            <el-row :gutter="24">
              <el-col :span="15">
                <div class="grid-content ep-bg-purple">设备寿命预测</div>
              </el-col>
              <el-col :span="1" :offset="8">
                <div class="grid-content ep-bg-purple-light">
                  <el-dropdown class="header-dropdown">
                    <span class="el-dropdown-link component" >
                        <el-avatar :src=avatarUrl></el-avatar>
                    </span>
                    <template #dropdown>
                      <el-dropdown-menu>
                        <el-dropdown-item @click="">个人信息修改</el-dropdown-item>
                        <el-dropdown-item @click="logout">用户登出</el-dropdown-item>
                      </el-dropdown-menu>
                    </template>
                  </el-dropdown>
                </div>
              </el-col>
            </el-row>
          </div>
        </el-affix>
      </el-header>
      <el-container>
        <el-aside width="200px">
          <el-affix :offset="103">
            <Menu></Menu>
          </el-affix>
        </el-aside>
        <el-container>
          <el-main>
            <router-view/>
          </el-main>
        </el-container>
      </el-container>
    </el-container>
  </div>
</template>

<style>
/* 头部样式 */
.el-header {
  height: 100px;
}

.system-header {
  background: linear-gradient(45deg, #00b0e8, #3b5998, #8b9dc3);
  color: white;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
  border-radius: 10px;
  font-size: 36px;
  font-weight: bold;
  letter-spacing: 1px;
  height: 60px;
  margin-left: -22px;
  margin-right: -22px;
  padding: 20px;
}

/* 侧边栏样式 */
.el-aside {
  background: linear-gradient(to right, #85D8CE, #6FA8DC);
  color: white;
  box-shadow: 4px 0 20px rgba(0, 0, 0, 0.2);
}

/* 彩虹渐变文字效果 */
.el-aside h2 {
  background-clip: text;
  -webkit-background-clip: text;
  color: transparent;
  background-image: linear-gradient(45deg, #ff00a9, #00eaff);
}

.el-header h1 {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  /* 根据项目需要选择字体 */
  font-size: 36px;
  font-weight: bold;
  letter-spacing: 1px;
  line-height: 1.5;
  /* 行高 */
  margin: 0;
  /* 去掉默认的外边距 */
}

/* 彩虹渐变边框效果 */
.el-header,
.el-aside {
  border: 2px solid transparent;
  background-clip: padding-box;
  -webkit-background-clip: padding-box;
  transition: border-color 0.3s ease;
}

.el-header:hover,
.el-aside:hover {
  border-color: #f39c12;
}

.el-main {
  /*background: red;*/
}

.el-footer {
  /*background: yellow;*/
}
.component:focus {
  /* 取消选中特效，可以根据需要修改样式 */
  outline: none;
}
</style>
<!-- <template>
  <div style="height: 90%">
    <div style="display:flex; right: 0;justify-content: flex-end; top: 0; z-index: 1;">
      <el-button-group>
        <el-button type="primary" round @click="importDataset">导入数据集</el-button>
        <el-button type="primary" round @click="getdataMsg">刷新数据表</el-button>
      </el-button-group>
    </div>
    <el-auto-resizer>
      <template #default="{ height, width }">
        <el-table-v2 :data="dataMsg" :width="width"
                     :height="height"
                     :columns="columns"
                     :fixed="true">
        </el-table-v2>
      </template>
    </el-auto-resizer>
  </div>
  <el-dialog
      v-model="dialogVisible"
      title="导入数据"
      width="30%"
      :before-close="handleClose"
  >
```

```
<el-card class="box-card">
  <el-form
      :ref="formRef"
      :model="uploadForm"
      :rules="uploadRules"
      label-width="100px"
      size="default"
  >
    <el-form-item label="数据集名称" prop="name">
      <el-input v-model="uploadForm.name" placeholder="请输入数据集名称"></el-input>
    </el-form-item>
    <el-form-item label="机器名称" prop="mechineName">
      <el-input v-model="uploadForm.mechineName" placeholder="请输入机器名称"></el-input>
    </el-form-item>
    <el-form-item label="组件" prop="component">
      <el-select v-model="uploadForm.component" placeholder="请选择组件" style="width: 115px">
        <el-option
            v-for="item in ComponentOptions"
            :key="item.value"
            :label="item.label"
            :value="item.value"
        >
        </el-option>
      </el-select>
    </el-form-item>
    <el-form-item label="负责人" prop="personInCharge">
      <el-input v-model="uploadForm.personInCharge" placeholder="请输入负责人"></el-input>
    </el-form-item>
  </el-form>
</el-card>
<el-upload drag accept=".xlsx" :auto-upload="false" ref="uploadRef" :limit="1" v-model:file-list="fileList">
  <template #trigger>
    <el-icon class="el-icon--upload">
      <upload-filled/>
    </el-icon>
    <div class="el-upload__text">
      拖拽到此处或 <em>点击此处上传</em>
    </div>
  </template>
```

```
      <template #tip>
        <div class="el-upload__tip">
          * 数据需要符合相应格式，否则无法处理
        </div>
      </template>
    </el-upload>
    <template #footer>
      <span class="dialog-footer">
        <el-button @click="resetUploadForm">取消</el-button>
        <el-button type="primary" @click="submitUpload">
          上传
        </el-button>
      </span>
    </template>
  </el-dialog>
  <el-dialog
      v-model="deleteVisible"
      title="删除数据"
      width="30%"
      :before-close="handleClose"
  >
    <el-card class="box-card" shadow="hover">
      <p>确定删除 id 为{{deleteRowData}} 的数据?</p>
    </el-card>
    <span slot="footer" class="dialog-footer">
      <el-button @click="deleteVisible = false">取 消</el-button>
      <el-button type="primary" @click="deleteRequest">确 定</el-button>
    </span>
  </el-dialog>
</template>
<script lang="tsx" setup>
import {computed, onMounted, ref} from 'vue'
import type {UploadUserFile} from 'element-plus'
import {ElAutoResizer, ElMessage, FormInstance, UploadInstance,ElButton} from 'element-plus'
import axiosInstance from "../../axios";

const uploadRef = ref<UploadInstance>()
const fileList = ref<UploadUserFile[]>([])
const dialogVisible = ref(false)
const deleteVisible = ref(false)
```

```
let formRef = ref<FormInstance | null>(null)
const componentSelect = ref('')
const deleteRowData = ref('')
const uploadForm = ref({
  name: '',
  mechineName: '',
  component: '',
  personInCharge: '',
})
const uploadRules = ref({
  name: [
    {required: true, message: '请输入数据集名称', trigger: 'blur'},
  ],
  mechineName: [
    {required: true, message: '请输入机器名称', trigger: 'blur'},
  ],
  component: [
    {required: true, message: '请输入组件', trigger: 'blur'},
  ],
  personInCharge: [
    {required: true, message: '请输入负责人', trigger: 'blur'},
  ],
})

const dataMsg = ref<
    {
      filename: string
      machine_name: string
      component_name: string
      component_type: string
      owner: string
    }[]
>([])

const getdataMsg = async () => {
  const url = '/data/all_user'
  await axiosInstance.get(url).then((res) => {
    if (res.data.code === 200) {
      dataMsg.value = res.data.data
      console.log(dataMsg.value)
      return
    } else {
      ElMessage.error('获取数据失败: ' + res.data.message)
      console.log(res.data)
      return
    }
  })
}
const generateColumnsFromItem = (item: any) => {
  return Object.keys(item).map(key => ({
    key: key,
    dataKey: key,
    title: key.charAt(0).toUpperCase() + key.slice(1).replace(/_/g, ' '), // 格式化标签
    width: 170,
  }));
}
const generateColumns = () => {
  if (dataMsg.value.length === 0) return [];
  // 使用数据的第一项来确定列
  const item = dataMsg.value[0];
  // 添加一个按扭列，用于预测
  const predictColumn = {
    key: 'operation',
    title: '操作',
    width: 160,
    cellRenderer: ({rowData}) => (
      <>
        <ElButton type="success" round  onClick={() => predict(rowData.id)}>预测</ElButton>
        <ElButton type="danger" round onClick={() => deleteRow(rowData.id)}>删除</ElButton>
      </>
    ),
  };
  return [
    ...generateColumnsFromItem(item),
    predictColumn,
  ];
}

const importDataset = () => {
  dialogVisible.value = true
```

```
}
const handleClose = (done: any) => {
  resetUploadForm()
  done()
}

const submitUpload = () => {
  if (uploadForm.value.component === '' || uploadForm.value.mechineName === ''
      || uploadForm.value.name === '' || uploadForm.value.personInCharge === ''
      || fileList.value.length === 0) {
    ElMessage({
      message: '请检查输入表单输入',
      type: 'error'
    })
    resetUploadForm()
    return
  } else {
    const file = fileList.value[0].raw

    const formData = new FormData()
    formData.append('file',file as File)
    formData.append('name', uploadForm.value.name)
    formData.append('component', uploadForm.value.component)
    const url = '/data/'
    axiosInstance.post(url, formData).then((res) => {
      if (res.data.code === 200) {
        ElMessage.success('上传成功')
        resetUploadForm()
      } else {
        ElMessage.error('上传失败: ' + res.data.message)
        resetUploadForm()
      }
    })

  }
}
const resetUploadForm = () => {
  uploadForm.value.component = ''
  uploadForm.value.mechineName = ''
  uploadForm.value.name = ''
  uploadForm.value.personInCharge = ''
  componentSelect.value = ''
  fileList.value = []
  dialogVisible.value = false
  deleteVisible.value = false
}
const predict = async (rowData: any) => {
  console.log("row: ", rowData);
  return
}
const deleteRow = async (rowData: any) => {
  console.log("row: ", rowData);
  deleteRowData.value = rowData
  deleteVisible.value = true
  return
}
const deleteRequest = async () => {
  const url ="/data/"+deleteRowData.value
  await axiosInstance.delete(url).then((res) => {
    if (res.data.code === 200) {
      ElMessage.success('删除成功')
      getdataMsg()
      deleteVisible.value = false
    } else {
      ElMessage.error('删除失败: ' + res.data.message)
      console.log(res.data)
    }
  })
}

// 挂载函数
onMounted(async () => {
  await getdataMsg();
  await getComponentOptions()
});

// 计算属性，用于根据数据动态生成列
const columns = computed(() => generateColumns());

const getComponentOptions = async () => {
  const url = '/component/all_user'
  await axiosInstance.get(url).then((res) => {
    if (res.data.code === 200) {
      let componentOptions: any[] = []
      res.data.data.map((item: any) => {
        componentOptions.push({
          value: item.id,
          label: item.name
        })
      })
      ComponentOptions = componentOptions
      console.log(ComponentOptions)
    } else {
      ElMessage.error('获取组件失败: ' + res.data.message)
      ComponentOptions = options
    }
  })
}
let ComponentOptions: any[] = []
const options = [
  {
    value: 'Option1',
    label: 'Option1',
  },
  {
    value: 'Option2',
    label: 'Option2',
  },
  {
    value: 'Option3',
    label: 'Option3',
  },
  {
    value: 'Option4',
    label: 'Option4',
  },
  {
    value: 'Option5',
    label: 'Option5',
  },
]
</script> -->

<template>
  <el-table :data="tableData">
    <el-table-column prop="time" label="时间"></el-table-column>

    <el-table-column prop="componentName" label="组件名称"></el-table-column>
    <el-table-column prop="remainingLife" label="剩余寿命/mins"></el-table-column>
  </el-table>
</template>

<script lang="ts" setup>
import { ref, onMounted } from 'vue';

interface TableData {
  time: string;
  componentId: number;
  componentName: string;

  remainingLife: number;
}

const tableData = ref<TableData[]>([]);
//给 tableData 赋测试值
tableData.value = [
  { time: '2023-11-01-13:20', componentName: 'Engine',componentId: 2, remainingLife: 1021 },
  {
    time: '2024-03-02-14:21', componentName: 'Lathe', componentId: 2, remainingLife: 2024
  },
  {
    time: '2024-03-23-21:30', componentName: 'Controller',componentId:6, remainingLife: 376
  },
];

onMounted(async () => {
      // const response = await axios.get('/api/components');
      // console.log("数据："+response.data);
  // tableData.value = response.data;
    });
```

```
</script>
<!--
<template>
  <el-table v-loading="loading" :data="tableData" style="width: 100%">
    <el-table-column prop="date" label="Date" width="180" />
    <el-table-column prop="name" label="Name" width="180" />
    <el-table-column prop="address" label="Address" />
  </el-table>
</template>

<script lang="ts" setup>
import { ref } from 'vue'

const loading = ref(true)

const tableData = [
  {
    date: '2016-05-02',
    name: 'John Smith',
    address: 'No.1518,  Jinshajiang Road, Putuo District',
  },
  {
    date: '2016-05-04',
    name: 'John Smith',
    address: 'No.1518,  Jinshajiang Road, Putuo District',
  },
  {
    date: '2016-05-01',
    name: 'John Smith',
    address: 'No.1518,  Jinshajiang Road, Putuo District',
  },
]
</script>

<style>
body {
  margin: 0;
}
.example-showcase .el-loading-mask {
  z-index: 9;
}
</style> -->

<template>
  <div>
    <el-row>
      <el-col :span="12">
        <el-form
            :model="componentForm"
            :rules="formRules"
            ref="componentFormRef"
            label-width="100px"
            class="demo-ruleForm"
        >
          <el-form-item label="组件名称" prop="name">
            <el-input v-model="componentForm.name" class="input-width"></el-input>
          </el-form-item>

          <el-form-item label="组件状态" prop="status">
            <el-select v-model="componentForm.status" placeholder="请选择" class="select-width">
              <el-option label="崭新" value=0 />
              <el-option label="良好" value=1 />
              <el-option label="一般" value=2 />
              <el-option label="破损" value=3 />
            </el-select>
          </el-form-item>

          <el-form-item label="组件描述" prop="description">
            <el-input v-model="componentForm.description" class="input-width"></el-input>
          </el-form-item>

          <el-form-item label="组件位置" prop="location">
            <el-input v-model="componentForm.location" class="input-width"></el-input>
          </el-form-item>

          <el-form-item label="模型" prop="model">
            <el-select  v-model="componentForm.modelId" placeholder="请选择" class="select-width" @visible-
change="getComponentOptions">
              <el-option
              v-for="item in ModelOptions"
              :key="item.id"
              :label="item.name"
              :value="item.id"
            />
          </el-select>
        </el-form-item>

        <el-form-item label="组件图片" prop="pic">
          <el-upload
            class="upload-demo"
            :onSuccess="handleAvatarSuccess"
            :beforeUpload="beforeAvatarUpload"
            :show-file-list="false"
            :onRemove="handleRemove"
          >
            <el-image
              style="width: 200px; height: 200px"
              :src="picUrl"
              fit="fill"
            >
              <template #placeholder>
                <div style="width: 200px; height: 200px; line-height: 200px; text-align: center;">点击上传</div>
              </template>
              <template #error>
                <div style="width: 200px; height: 200px; line-height: 200px; text-align: center;
background:lightgrey">点击上传</div>
              </template>
            </el-image>
          </el-upload>
        </el-form-item>

        <el-form-item>
          <el-button type="primary" @click="submitForm">提交</el-button>
          <el-button @click="resetForm">重置</el-button>
        </el-form-item>
      </el-form>
    </el-col>

    <el-col :span="12">
```

```
        <el-collapse v-model="componentForm.modelId">
          <el-collapse-item title="Model Details" :name="componentForm.modelId">
            <div v-for="item in selectedModel" :key="item.id">
              <p>id: {{ item.id }}</p>
              <p>name: {{ item.name }}</p>
              <p>style: {{ item.style }}</p>
              <p>status: {{ item.status }}</p>
              <p>description: {{ item.description }}</p>
              <p>uploaded_time: {{ item.uploaded_time }}</p>
            </div>
          </el-collapse-item>
        </el-collapse>
      </el-col>
    </el-row>
  </div>
</template>

<style scoped>
.input-width {
    width: 300px;
}
.select-width {
    width: 100px;
}
.option-input {
  width: 100%;
  margin-bottom: 8px;
}
</style>

<script lang="ts" setup>
import { ref ,onMounted,watch} from 'vue'
import {ElMessage} from 'element-plus'
import axiosInstance from "../../axios.ts";

interface ComponentInfor {
  name: string
  status: string
  description: string
  location: string
  modelId: number | null
  life_forecast: number | string
  pic: File
}
interface  ModelInf {
  id: number
  name: string
  style: string
  status: string
  description: string
  uploaded_time: string
  md5: string
}
const picUrl = ref('')
const componentForm = ref<ComponentInfor>({
  name: '',
  status: '',
  description: '',
  location: '',
  modelId: null, //kongzhi
  life_forecast: '',
  pic : new File([''], 'filename')
})
const selectedModel = ref<ModelInf[]>([])

//监视函数当 model 改变时触发，将选择的 model 信息赋值给 selectedModel
watch(() => componentForm.value.modelId, (val:any) => {
  selectedModel.value = ModelOptions.value.filter((item) => item.id === val)
})

const formRules = {
  name: [
    { required: true, message: '请输入组件名称', trigger: 'blur' }
  ],
  status: [
    { required: true, message: '请选择组件状态', trigger: 'change' }
  ],
  description: [
    { required: true, message: '请输入组件描述', trigger: 'blur' }
  ],
  location: [
    { required: true, message: '请输入组件位置', trigger: 'blur' }
  ],
  modelId: [
    { required: true, message: '请选择模型', trigger: 'change' }
  ],
  life_forecast: [
    { required: true, message: '请输入寿命预测', trigger: 'blur' }
  ],
  pic: [
    { required: true, message: '请上传组件图片', trigger: 'change' }
  ]
}

let ModelOptions= ref<ModelInf[]>([])
const defaultModelOpt = [
  {
    id: null,
    name: 'model01',
    style: 'A123',
    status: 'active',
    description: 'this is model01',
    uploaded_time: '2021-10-01',
    md5: '123456'
  },
  {
    id: 1,
    name: 'model02',
    style: 'B456',
    status: 'active',
    description: 'this is model02',
    uploaded_time: '2021-10-02',
    md5: '123456'
  },
  {
    id: 2,
    name: 'model03',
    style: 'C789',
    status: 'active',
    description: 'this is model03',
    uploaded_time: '2021-10-03',
    md5: '123456'
  }
]

async function submitForm() {
  ElMessage.success('添加组件')
  console.log(componentForm.value)
  const formData = new FormData()
  formData.append('name', componentForm.value.name)
  if(componentForm.value.status !== null && componentForm.value.status !== undefined){
    formData.append('status', componentForm.value.status.toString())
  }
  formData.append('description', componentForm.value.description)
  formData.append('location', componentForm.value.location)
  //@ts-ignore
  formData.append('model', componentForm.value.modelId)
  formData.append('pic', componentForm.value.pic)

  await axiosInstance.post('/component/upload', formData).then((res) => {
    if (res.data.code === 200) {
      ElMessage.success('添加组件成功')
      resetForm()
    } else {
      console.log('添加组件失败: ' + res.data.message)
      ElMessage.error('添加组件失败: ' + res.data.message)
    }
  })
}
const resetForm = async () => {
  console.log('reset')
  componentForm.value.name = ''
  componentForm.value.status = ''
  componentForm.value.description = ''
  componentForm.value.location = ''
  componentForm.value.modelId = null
  componentForm.value.life_forecast = ''
  componentForm.value.pic = new File([''], 'filename')
  picUrl.value = ''
};
```

```
const handleAvatarSuccess = (_: any, file: any) => {
  console.log("Ok!")
  picUrl.value = URL.createObjectURL(file.raw)
}

const beforeAvatarUpload = (file: any) => {
  const isJPG = file.type === 'image/jpeg'
  const isPNG = file.type === 'image/png'
  const isGIF = file.type === 'image/gif'
  const isLt6M = file.size / 1024 / 1024 < 6

  if (!isJPG && !isPNG && !isGIF) {
    ElMessage.error('上传头像图片只能是 JPG/PNG/GIF 格式!')
  }
  if (!isLt6M) {
    ElMessage.error('上传头像图片大小不能超过 6MB!')
  }
  if((isJPG || isPNG || isGIF) && isLt6M){
    const render = new FileReader();
    render.onload = (e) => {
      picUrl.value = e.target?.result as string
    }
    render.readAsDataURL(file)
    console.log("src: ",picUrl.value)
    componentForm.value.pic = file
    return false;
  }
  return false;
}
const handleRemove = () => {
  picUrl.value = ''
}

const getComponentOptions =  () => {
  const url = '/model/all'
  axiosInstance.get(url).then((res) => {
    if (res.data.code === 200) {
      ModelOptions.value = res.data.data
      console.log("模型信息"+ModelOptions.value)
    } else {
      ElMessage.error('获取组件失败: ' + res.data.message)
      ModelOptions.value = defaultModelOpt
    }
  })
}

onMounted(async () => {
  getComponentOptions()
});
</script>

<template>
  <div style="display: flex; justify-content: flex-end; margin-bottom: 20px;">
    <el-button type="primary" @click="add_component">添加组件</el-button>
  </div>

  <el-space direction="vertical" alignment="flex-start" style="margin-bottom: 20px;">
    <el-skeleton style="width: 240px" :loading=false animated :count="3">

      <template #default>
        <el-row :gutter="20">
          <el-col :span="8" v-for="item in lists" :key="item.name">
            <el-card :body-style="{ padding: '0px', marginBottom: '1px' }">
              <img :src="item.imgUrl" class="image multi-content" style="height: 250px; object-fit: contain;" />
              <div style="padding: 14px">
                <span>{{ item.name }}</span>
                <div class="bottom card-header">
                  <el-countdown title="life forecast" :value="item.life_forecast" />
                  <el-button text class="button" bg @click="handleEdit(item)">编辑</el-button>
                  <el-button text class="button" type="primary" @click="handlePredict(item)" bg>预测</el-button>
                </div>
              </div>
            </el-card>
          </el-col>
        </el-row>
      </template>

    </el-skeleton>
  </el-space>
```

```
<el-drawer
  v-model="drawer"
  :title="currentItem?.name"
  direction="rtl"

>
  <el-form label-width="120px">
    <el-form-item label="ID">
      <el-input disabled v-model="currentItem.id"></el-input>
    </el-form-item>

    <el-form-item label="Life Forecast">
      <el-input disabled v-model="currentItem.life_forecast" />
    </el-form-item>

    <el-form-item label="Location">
      <el-input v-model="uploadItem.location" />
    </el-form-item>
    <el-form-item label="Status">
      <el-input v-model="uploadItem.status" />
    </el-form-item>

    <el-form-item label="Name">
      <el-input v-model="uploadItem.name" />
    </el-form-item>
    <el-form-item label="Model">
      <!-- 使用 modelList，显示当前 model 的 name -->
      <el-select v-model="uploadItem.model__name" placeholder="请选择" class="select-width">
        <el-option
          v-for="item in modelList"
          :key="item.model__id"
          :label="item.model__name"
          :value="item.model__id"
        />
      </el-select>
    </el-form-item>

    <el-form-item label="Image">
      <img :src="currentItem.imgUrl" style="max-width: 200px; width: auto; height: auto;" />
```

```
        <el-upload action="https://jsonplaceholder.typicode.com/posts/">
          <el-button slot="trigger" size="small" type="primary">选择照片</el-button>
        </el-upload>
      </el-form-item>

      <el-form-item style="display: flex; justify-content: space-between;">
        <el-button type="danger" @click="handleRemove()">移除</el-button>
        <el-button type="primary" @click="drawer = false">Save</el-button>
      </el-form-item>
    </el-form>

  </el-drawer>
  <el-dialog v-model="dialogVisible"  title="Predict">
    <el-upload
      class="upload-demo"
      drag
      action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"
      multiple
    >
      <el-icon class="el-icon--upload"><upload-filled /></el-icon>
      <div class="el-upload__text">
        Drop file here or <em>click to upload</em>
      </div>
    </el-upload>
    <span slot="footer" class="dialog-footer">
      <el-button @click="dialogVisible = false">Cancel</el-button>
      <el-button type="primary" @click="Predict">Confirm</el-button>
    </span>
  </el-dialog>

</template>

<script lang="ts" setup>
import axiosInstance from "../../axios.ts";
import { onMounted, ref } from 'vue'
import { ElMessageBox } from 'element-plus'
import { UploadFilled } from '@element-plus/icons-vue'
import { useRouter } from 'vue-router';

const router = useRouter();
```

```
interface ListItem {
  id: number
  imgUrl: string
  name: string
  model__id: number
  model__name: string
  life_forecast: number
  updated_time: string
  location: string
  status: string
}
interface model{
  model__name: string
  model__id: number
}

interface uploadInfor{
  component_id:number
  name:string
  status: string
  location:string
  description:string
  imgUrl:string
  model__id: number
  model__name: string
}

const dialogVisible = ref(false);
const drawer = ref(false)
const loading = ref(true)
const lists = ref<ListItem[]>([])
const currentItem = ref<ListItem | null>(null);
const uploadItem = ref<uploadInfor | null>(null);

function add_component() {
  router.push({ name: 'component-add' });
}

//model 数组
const modelList = ref<model[]>([])
```

```
//预测函数
const handlePredict = (item: ListItem) => {
  dialogVisible.value = true;
  currentItem.value = item;
}

const Predict = () => {

  //正在预测
  ElMessageBox.alert('预测成功')
  currentItem.value!.life_forecast = Date.now() + Math.random() * 1000 * 60 * 60 * 24;
  dialogVisible.value = false;
};
const handleRemove = () => {
  ElMessageBox.confirm('Are you sure you want to remove this?')
    .then(() => {
      lists.value = lists.value.filter((item) => item.id !== currentItem.value?.id)
      const url = '/component/delete/' + currentItem.value?.id
      axiosInstance.delete(url).then(() => {
        drawer.value = false
        ElMessageBox.alert('删除成功')
      })
        .catch(() => {
          // catch error
          ElMessageBox.alert('删除失败')
        })
    })
    .catch(() => {
      // catch error
    })
}

const handleEdit = (item: ListItem) => {
  drawer.value = true;
  currentItem.value = item;
  uploadItem.value = {
    component_id: item.id,
    name: item.name,
    status: item.status,
    location: item.location,
    description: item.model__name,
    imgUrl: item.imgUrl,
    model__id: item.model__id,
    model__name: item.model__name
  }
};

//获取所有组件信息，/component/all_user
const getComponentList = async () => {
  const url = '/component/all_user'
  axiosInstance.get(url).then((res) => {
    console.log("组件信息"+res.data.data)
    lists.value = res.data.data
    //给每个 lists 的 life_forecast 本身值加上 Date.now()
    lists.value.forEach((item) => {
      item.life_forecast = Date.now() + item.life_forecast
    })
    //会返回 model__name 和 model__id,赋值给 modelList,
    modelList.value = res.data.data.map((item: ListItem) => {
      return {
        model__name: item.model__name,
        model__id: item.model__id
      }
    })
    //去重
    modelList.value = modelList.value.filter((item, index, self) =>
      index === self.findIndex((t) => (
        t.model__id === item.model__id
      ))
    )
    lists.value.forEach((item) => {
       getComponentPic(item.id)
    })
  }).catch((err) => {
    console.log(err)
  })
}

const getComponentPic = async (id: number) => {
  const url = '/component/pic/'+id
  axiosInstance.get(url, { responseType: 'blob' }).then((res) => {
    lists.value.forEach((item) => {
    const blob = new Blob([res.data], { type: res.headers['content-type'] });
    if(item.id === id){
        item.imgUrl = URL.createObjectURL(blob);
      }
    })
  }).catch((err) => {
    console.log(err)
  })
}


onMounted(() => {
  loading.value = false
  getComponentList()
})
</script>
<style scoped>
.image {
  width: 100%;
  height: auto;
}
body {
  margin: 0;
}
.example-showcase .el-loading-mask {
  z-index: 9;
}
</style>

<template>
  <el-form ref="modelInfor" :model="modelInf" label-width="120px">
    <el-form-item label="模型名称" prop="name">
      <el-input class='input-width'
                autosize
                type="textarea"
                v-model="modelInf.name" placeholder="Model Name"/>
    </el-form-item>
    <el-form-item label="简述" prop="description">
      <el-input autosize
                type="textarea"
                v-model="modelInf.description"
                placeholder="Description"/>
    </el-form-item>
    <el-form-item label="状态" prop="status">
      <el-select class="select-width" v-model="modelInf.status" placeholder="Status">
        <el-option label="启用" value="1"/>
        <el-option label="禁用" value="0"/>
      </el-select>
    </el-form-item>

    <el-form-item label="模型风格" prop="style">
      <el-select v-model="modelInf.style" placeholder="Select" style="width: 240px">
        <el-option
            v-for="item in modelStyle"
            :key="item.value"
            :label="item.label"
            :value="item.value"
        />
        <template #footer>
          <el-button v-if="!isAdding" text bg size="small" @click="onAddOption">
            Add an option
          </el-button>
          <template v-else>
            <el-input
                v-model="optionName"
                class="option-input"
                placeholder="input option name"
                size="small"
            />
            <el-button type="primary" size="small" @click="onConfirm">
              confirm
            </el-button>
            <el-button size="small" @click="clear">cancel</el-button>
          </template>
        </template>
      </el-select>
    </el-form-item>
```

```
    <el-form-item label="上传模型">
      <el-upload
          :on-success="handleModelSuccess"
          :before-upload="beforeModelUpload"
          :on-remove="handleRemove"
          drag
          :show-file-list="true"
      >
        <el-icon class="el-icon--upload">
          <upload-filled/>
        </el-icon>
        <div class="el-upload__text">
          Drop file here or <em>click to upload</em>
        </div>
        <template #tip>
          <div class="el-upload__tip">
            choose the model file
          </div>
        </template>
      </el-upload>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="submitForm(modelInf)">Submit</el-button>
      <el-button @click="resetForm">Cancel</el-button>
    </el-form-item>
  </el-form>

</template>

<style scoped>
.input-width {
  width: 300px;
}

.select-width {
  width: 100px;
}

.option-input {
  width: 100%;
  margin-bottom: 8px;
}
</style>

<script lang="ts" setup>
import {ref} from 'vue'
import {useRouter} from 'vue-router';
import {CheckboxValueType, ElMessage} from 'element-plus'
import axiosInstance from "../../axios";
const isAdding = ref(false)
const value = ref<CheckboxValueType[]>([])
const optionName = ref('')
const modelFile =ref()
const modelStyle = ref([
  {
    value: 'reinforcement-learning',
    label: '强化学习',
  },
  {
    value: 'CNN',
    label: 'CNN',
  },

])
const router = useRouter();
const resetForm = () => {
  console.log('reset')

  router.go(-1);
};

const onAddOption = () => {
  isAdding.value = true
}

const onConfirm = () => {
  if (optionName.value) {
    modelStyle.value.push({
      label: optionName.value,
      value: optionName.value,
    })
    clear()
  }
}

const clear = () => {
  optionName.value = ''
  isAdding.value = false
}

const modelInf = ref({
  name: '',
  style: '',
  description: '',
  status: '',
})

const submitForm = (modelInf: any) => {
  const url = '/model/'
  const formData = new FormData()
  formData.append('file', modelFile.value as File)
  formData.append('name', modelInf.name)
  formData.append('style', modelInf.style)
  formData.append('status', modelInf.status)
  formData.append('description', modelInf.description)
  for (let [key, value] of formData.entries()) {
    console.log(key, value);
  }
  axiosInstance.post(url, formData).then((res) => {
    if (res.data.code === 200) {
      ElMessage.success('上传成功')
      resetForm()
    } else {
      ElMessage.error('上传失败: ' + res.data.message)
      console.log(res.data)
    }
  })
}
const handleModelSuccess = (res: any, file: any) => {
  console.log("Ok!")
```

```
}
const beforeModelUpload = (file: any) => {
  console.log(file)
  modelFile.value = file
  return false
}
const handleRemove = (file: any, fileList: any) => {
  console.log(file, fileList)
}

</script>

<template>
  <div style="float: right;">
    <el-button type="primary" @click="add_model">添加模型</el-button>
    <el-button type="primary" @click="getModelData">刷新</el-button>
  </div>

  <el-table :data="pagedData">
    <el-table-column prop="id" label="模型编号"></el-table-column>
    <el-table-column prop="name" label="模型名称"></el-table-column>
    <el-table-column prop="style" label="模型风格"></el-table-column>
    <el-table-column prop="status" label="模型状态"></el-table-column>
    <el-table-column prop="description" label="模型描述"></el-table-column>
    <el-table-column prop="uploaded_time" label="创建时间"></el-table-column>
    <el-table-column label="操作">
      <template #default="scope">
        <el-button type="success" @click="handleEdit(scope.$index, scope.row)">修改</el-button>
        <el-button type="danger" @click="handleDelete(scope.$index, scope.row)">删除</el-button>
      </template>
    </el-table-column>
  </el-table>
  <el-dialog title="修改模型信息" v-model="dialogVisible" width="40%" :before-close="handleClose">
    <el-form :model="modelForm" :rules="formRules" ref="formRef" label-width="100px">
      <el-form-item label="模型编号" prop="id">
        <el-input v-model="modelForm.id" disabled></el-input>
      </el-form-item>
      <el-form-item label="模型名称" prop="name">
        <el-input v-model="modelForm.name"></el-input>
      </el-form-item>
      <el-form-item label="模型风格" prop="style">
        <el-input v-model="modelForm.style"></el-input>
      </el-form-item>
      <el-form-item label="模型状态" prop="status">
        <el-input v-model="modelForm.status"></el-input>
      </el-form-item>
      <el-form-item label="模型描述" prop="description">
        <el-input v-model="modelForm.description"></el-input>
      </el-form-item>
      <el-form-item label="模型文件" prop="file">
        <el-upload drag accept=".zip" style="width: 100%;" :auto-upload="false" ref="uploadRef" :limit="1" v-
model:file-list="fileList">
          <template #trigger>
            <el-icon class="el-icon--upload">
              <upload-filled/>
            </el-icon>
            <div class="el-upload__text">
              拖拽到此处或 <em>点击此处上传</em>
            </div>
          </template>
          <template #tip>
            <div class="el-upload__tip">
              * 上传模型文件
            </div>
          </template>
        </el-upload>
      </el-form-item>

    </el-form>
    <span slot="footer" class="dialog-footer">
      <el-button @click="dialogVisible = false">取 消</el-button>
      <el-button type="primary" @click="dialogVisible = false">确 定</el-button>
    </span>
  </el-dialog>

  <el-dialog title="删除模型" v-model="deleteVisible" width="30%" :before-close="handleClose">
    <el-card class="box-card">
      <p>确定删除 id 为{{modelForm.id}} 的模型?</p>
    </el-card>
    <span slot="footer" class="dialog-footer">
      <el-button @click="deleteVisible = false">取 消</el-button>
      <el-button type="primary" @click="deleteRow">确 定</el-button>
    </span>
  </el-dialog>
  <el-pagination
      @size-change="handleSizeChange"
      @current-change="handleCurrentChange"
      :current-page="currentPage"
      :page-size="pageSize"
      :hide-on-single-page="true"
      layout="total, prev, pager, next, jumper"
      :total="tableData.length">
  </el-pagination>
</template>

<script setup lang="ts">
import {useRouter} from 'vue-router';
import {ref,onMounted} from 'vue'
import axiosInstance from "../../axios";
import {ElMessage, type FormInstance, type UploadUserFile} from "element-plus";
//改成组合式
const tableData = ref([]);
const currentPage = ref(1);
const pageSize = ref(10);
const pagedData = ref([]);
const router = useRouter();
const dialogVisible = ref(false);
const deleteVisible = ref(false);
const fileList = ref<UploadUserFile[]>([])
const modelForm = ref({
  name: '',
  style: '',
  status: '',
  description: '',
  file: '',
  id: ''
});
const formRules = ref({
  name: [
    {required: true, message: '请输入模型名称', trigger: 'blur'}
  ],
  style: [
    {required: true, message: '请输入模型风格', trigger: 'blur'}
  ],
  status: [
    {required: true, message: '请输入模型状态', trigger: 'blur'}
  ],
  description: [
    {required: true, message: '请输入模型描述', trigger: 'blur'}
  ],
});
const formRef = ref<FormInstance | null>(null)

function handleSizeChange(val: any) {
  pageSize.value = val;
  updatePagedData();
}

function handleCurrentChange(val: any) {
  currentPage.value = val;
  updatePagedData();
}

function updatePagedData() {
  const start = (currentPage.value - 1) * pageSize.value;
  const end = start + pageSize.value;
  pagedData.value = tableData.value.slice(start, end);
}

function add_model() {
  router.push({name: 'model-add'});
}

const getModelData = async () => {
  const url = '/model/all'
  await axiosInstance.get(url).then((res) => {
    if (res.data.code === 200) {
      tableData.value = res.data.data
      updatePagedData()
      return
    } else {
      ElMessage.error('获取模型失败: ' + res.data.message)
      return
    }
  })
};

const handleEdit = (index: number, row: any) => {
  console.log(index, row);
  modelForm.value = row;
  dialogVisible.value = true;
};
onMounted(async () => {
  await getModelData()
})
const handleClose = (done: any) => {
  deleteVisible.value = false;
  dialogVisible.value = false;
  done();
};

const handleDelete = async (index: number, row: any) => {
  console.log(index, row);
  modelForm.value = row;
  deleteVisible.value = true;
};
const updateRow = async () => {
  if (!formRef.value) {
    ElMessage.error('请输入模型信息')
    return
  }
  await formRef.value?.validate((valid: any, _) => {
    if (valid) {
      const url = '/model/update'
      const data = {
        id: modelForm.value.id,
        name: modelForm.value.name,
        style: modelForm.value.style,
        status: modelForm.value.status,
        description: modelForm.value.description,
      }
      const headers = {
        'Content-Type': 'application/x-www-form-urlencoded'
      }
      axiosInstance.post(url, data, {headers}).then((res) => {
        if (res.data.code === 200) {
          ElMessage.success('修改成功')
          getModelData()
          dialogVisible.value = false
        } else {
          ElMessage.error('修改失败: ' + res.data.message)
          console.log(res.data)
        }
      })
    } else {
      ElMessage.error('修改失败，请检查模型信息')
    }
  })
}
const deleteRow = async () => {
  const url = '/model/'+modelForm.value.id
  await axiosInstance.delete(url).then((res) => {
    if (res.data.code === 200) {
      ElMessage.success('删除成功')
      getModelData()
      deleteVisible.value = false
    } else {
      ElMessage.error('删除失败: ' + res.data.message)
      console.log(res.data)
    }
  })
}
</script>
<script setup lang="ts">
import {CanvasRenderer} from 'echarts/renderers';
import {BarChart, PieChart} from 'echarts/charts';
import {use} from 'echarts/core';
import {ref, onMounted} from "vue";
import VChart from 'vue-echarts';
import axiosInstance from "../../axios";
import {ElMessage } from "element-plus";

use([BarChart, PieChart, CanvasRenderer]);

const jsonHeaders = {
  'Content-Type': 'application/json',
};
let pie1 = ref<any>([])
let pie2 = ref<any>([])
let line1 = ref<any>([])
let gauges = ref<any[]>([])

async function fetchPie1Data() {
  const res = axiosInstance.get("model/style", {headers: jsonHeaders});
  const response = await res
  if (response.status !== 200) {
    console.error('Error:', response.status, response.statusText);
    ElMessage.error('获取数据失败: ' + response.statusText);
    return;
  }
  if(response.data.code !== 200){
    console.error('Error:', response.data.message);
    ElMessage.error('获取数据失败: ' + response.data.message);
    return;
  }

  const data = await response.data.data
  let pieData = [];
  for (const key in data) {
    pieData.push({
      name: key,
      value: data[key].toFixed(2)
    });
  }
  pie1.value = {
    title: {
      text: '模型占比',
      left: 'center',
    },
    series: [
      {
        name: '模型',
        type: 'pie',
        radius: '50%',
        center: ['60%', '40%'],
        data: pieData
      }
    ],
    legend: {
      orient: 'vertical',
      left: 'left',
      data: pieData.map((name: { name: string }) => name.name),
    },
    emphasis: {
      itemStyle: {
        shadowBlur: 10,
        shadowOffsetX: 0,
        shadowColor: 'rgba(0, 0, 0, 0.5)',
      },
    },
    tooltip: {
      trigger: 'item',
      formatter: '{a} <br/>{b} : {c} ({d}%)',
    },
  }
}


async function fetchPie2Data() {
  const res = axiosInstance.get("/component/location_count", {headers: jsonHeaders});
  const response = await res;
  if (response.status !== 200) {
    console.error('Error:', response.status, response.statusText);
    ElMessage.error('获取数据失败: ' + response.statusText);
    return;
  }
  if(response.data.code !== 200){
    console.error('Error:', response.data.code, response.data.message);
    ElMessage.error('获取数据失败: ' + response.data.message);
    return;
  }
  const data = await response.data.data;
  let pieData = [];
  for (const key in data) {
    pieData.push({
      name: key,
      value: data[key]
    });
  }
  pie2.value = {
    title: {
      text: '各位置组件分布',
      left: 'center',
    },
    series: [
      {
        name: '应用',
        type: 'pie',
        radius: '50%',
        center: ['50%', '40%'],
        data: pieData,
      }
    ],
    legend: {
      orient: 'vertical',
      left: 'left',
      data: pieData.map((item: { name: string }) => item.name),
    },
    emphasis: {
      itemStyle: {
        shadowBlur: 10,
        shadowOffsetX: 0,
        shadowColor: 'rgba(0, 0, 0, 0.5)',
      },
    },
    tooltip: {
      trigger: 'item',
      formatter: '{a} <br/>{b} : {c} ({d}%)',
    },
  }
}

fetchPie1Data();
fetchPie2Data();

line1.value = {
  title: {
    text: '监控设备的数量'
  },
  tooltip: {},
  xAxis: {
    data: ['1 月', '2 月', '3 月', '4 月', '5 月', '6 月', '7 月', '8 月', '9 月', '10 月', '11 月', '12 月']
  },
  yAxis: {},
  series: [
    {
      name: '2022 年',
      type: 'line',
      data: Array.from({length: 12}, () => Math.floor(Math.random() * 100))
    },
    {
      name: '2023 年',
      type: 'line',
      data: Array.from({length: 12}, () => Math.floor(Math.random() * 100))
    }
  ]
}
for (let i = 0; i < 3; i++) {
  gauges.value[i] = {
    title: {
      text: '指标' + i,
      left: 'center',
    },
    series: [
      {
        name: '指标' + i,
        type: 'gauge',
        detail: {
          formatter: '{value}%',
          fontSize: 8  // 设置字体大小
        },
        data: [{value: (Math.random() * 100).toFixed(2)}]
      }
    ]
  }
}

onMounted(async () => {
  await fetchPie1Data();
  await fetchPie2Data();
  window.setInterval(fetchPie1Data, 60000);
});
</script>

<template>
  <el-row :gutter="20" style="width: 100%;height: 100%">
    <el-col :span="15">
      <div style="height: 40%;">
        <v-chart class="chart" :option="line1" autoresize/>
      </div>
      <el-row :gutter="20" style="height:60%;">
        <el-col :span="10">
          <v-chart class="chart" :option="pie1" autoresize/>
        </el-col>
        <el-col :span="10">
          <v-chart class="chart" :option="pie2" autoresize/>
        </el-col>
      </el-row>
    </el-col>
    <el-col :span="5" style="height: 70%;">
      <div style="height: 47%;" v-for="gauge in gauges">
        <v-chart class="chart" :option="gauge" autoresize/>
      </div>
    </el-col>
  </el-row>
</template>

<script setup lang="ts">
import {
  ElForm,
  ElFormItem,
  ElInput,
  ElCard,
  ElButton,
  ElMessage,
  ElAvatar,
  ElUpload,
  type UploadUserFile
} from "element-plus"
import type { FormInstance, FormRules } from 'element-plus'
import { ref } from 'vue'
import { useRouter } from 'vue-router'
import axiosInstance from "../../axios";
const avafile=ref()
const form = ref({
  username: '',
  password: '',
  email: '',
  phone: '',
  avatar: ''
})
const formRules = ref<FormRules>({
  username: [
    { required: true, message: '请输入用户名', trigger: 'blur' }
  ],
  password: [
    { required: true, message: '请输入密码', trigger: 'blur' }
  ],
  email: [
    { required: true, message: '请输入邮箱', trigger: 'blur' }
  ],
  phone: [
    { required: true, message: '请输入手机号', trigger: 'blur' }
  ],
  avatar: [
    { required: true, message: '请上传头像', trigger: 'blur' }
  ]
})
```

```
const formRef = ref<FormInstance | null>(null)
const router = useRouter()
const handleAvatarSuccess = (res: any, file: any) => {
  console.log("Ok!")
  form.value.avatar = URL.createObjectURL(file.raw)
}
const beforeAvatarUpload = (file: any) => {
  const isJPG = file.type === 'image/jpeg'
  const isPNG = file.type === 'image/png'
  const isGIF = file.type === 'image/gif'
  const isLt2M = file.size / 1024 / 1024 < 2

  if (!isJPG && !isPNG && !isGIF) {
    ElMessage.error('上传头像图片只能是 JPG/PNG/GIF 格式!')
  }
  if (!isLt2M) {
    ElMessage.error('上传头像图片大小不能超过 2MB!')
  }
  if((isJPG || isPNG || isGIF) && isLt2M){
    const render = new FileReader();
    render.onload = (e) => {
      form.value.avatar = e.target?.result as string
    }
    render.readAsDataURL(file)
    console.log("src: ",form.value.avatar)
    avafile.value = file
    return false;
  }
  return false;
}
const handleRemove = () => {
  form.value.avatar = ''
}
const register =async()=>{
  const url = '/user/register'
  const formData = new FormData()
  // 读取 src 文件
  const file = avafile.value
  console.log(avafile.value)
  formData.append('username', form.value.username)
  formData.append('password', form.value.password)
  formData.append('email', form.value.email)
  formData.append('phone', form.value.phone)
  formData.append('avatar', file as File)
  axiosInstance.post(url, formData).then((res) => {
    console.log(res.data)
    if (res.data.code === 200) {
      ElMessage.success('注册成功')
      router.push({
        path: "/login",
        replace: true
      });
    } else {
      ElMessage.error('注册失败: '+ res.data.message)
      console.log(res.data)
    }
  })
}
</script>
<template>
  <div>
    <el-card class="box-card" style="width: 500px;margin: 0 auto;">
      <el-form :model="form" :rules="formRules" ref="formRef" label-width="80px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="form.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
          <el-input v-model="form.password" type="password"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
          <el-input v-model="form.email"></el-input>
        </el-form-item>
        <el-form-item label="手机号" prop="phone">
          <el-input v-model="form.phone"></el-input>
        </el-form-item>
        <el-form-item label="头像" prop="avatar">
          <el-upload
              class="upload-demo"
              :onSuccess="handleAvatarSuccess"
              :beforeUpload="beforeAvatarUpload"
              :show-file-list="false"
              :onRemove="handleRemove"
          >
            <el-avatar :src="form.avatar" shape="circle" size="large">user</el-avatar>
          </el-upload>
        </el-form-item>
        <div class="btnGroup">
          <el-button type="primary" @click="register">注册</el-button>
          <router-link to="/login">
            <el-button style="margin-left:10px">登录</el-button>
          </router-link>
        </div>
      </el-form>
    </el-card>
  </div>

</template>
<script setup lang="ts">
import { onMounted } from 'vue';
import { useRouter } from "vue-router";
import 'element-plus/dist/index.css'
const router = useRouter();

onMounted(() => {
    router.push({
        path: "/login",
        replace: true
    });
});
</script>

<template>
    <router-view/>
</template>

<style scoped>
.logo {
    height: 6em;
    padding: 1.5em;
    will-change: filter;
  transition: filter 300ms;
}

.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}

.logo.vue:hover {
  filter: drop-shadow(0 0 2em #42b883aa);
}
</style>
```

后端部分代码

```
from openpyxl import load_workbook
import torch
import torchvision
from torch import nn
import copy
import numpy as np
import sys
import os

if len(sys.argv) < 2:
  print("Usage: python exc2npy.py <excelfile>")
  sys.exit(1)
# 第一个参数是脚本名称,第二个参数是文件名
excel_file = sys.argv[1]
# 分隔文件名和扩展名
file_name, ext = os.path.splitext(excel_file)
print(file_name, ext)

df = load_workbook(excel_file)
sheet=df.active
a=[]
for i in sheet.iter_rows():
  i_d=[cell.value for cell in i]
  a.append(i_d)
a=np.array(a)
aa = np.reshape(a,(a.shape[0]//18//3,3,18,18))
tbt = torch.tensor(aa)
```

```
model = torchvision.models.resnet50(pretrained = True)
model = model
for param in model.parameters():
    param.requires_grad = False
model_last = model.fc.in_features
model.conv1 = nn.Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(3, 3), bias=False)
model.fc = nn.Sequential(nn.Linear(model_last,128),
                         nn.ReLU(inplace = True),
                         nn.Linear(128,70))
loss_intitial = copy.deepcopy(model.cuda())
def test(tbt,model):
        images=tbt.cuda()
        images = images.float()
        outputs=model(images)
        _,predicted=torch.max(outputs.data,1)
        print(predicted)
model = torch.load(r'./model/model_resnet_1.pt')#加载模型
test(tbt,model)


#-*- coding : utf-8 -*-
# coding: utf-8

import sys
import os

if len (sys.argv) < 2:
    print("Usage: python random_life.py <excelfile>")
    sys.exit(1)

excel_file = sys.argv[1]
file_name, ext = os.path.splitext(excel_file)
# 生成随机整数，范围在 1~70
import random
print(random.randint(1,70),end='')

from fastapi import APIRouter, Form, UploadFile, Depends, HTTPException, Response
from database.models import MModel, User, Component
from tools.security import get_current_user
from typing import Union
from common.CommonResponse import CommonResponse

componentAPI = APIRouter()

@componentAPI.post("/upload", response_model_exclude={"pic", "model", "user"})
async def upload_component(
    pic: UploadFile,
    name: str = Form(...),
    location: str = Form(...),
    model: Union[int, None] = Form(...),
    description: Union[str, None] = Form(...),
    current_user: User = Depends(get_current_user)
):
    # 检查组件名是否重复
    if await Component.exists(name=name, user=current_user):
        return CommonResponse.error(110, "组件名已存在")

    if model is None:
        component = await Component.create(
            name=name,
            location=location,
            user=current_user,
            pic=await pic.read()
        )
    else:
        model = await MModel.get(id=model)
        if not model:
            return CommonResponse.error(103, "模型不存在")
        component = await Component.create(
            name=name,
            location=location,
            model=model,
            user=current_user,
            pic=await pic.read(),
            description=description
        )
    component.model = ''
    component.pic = ''
    return CommonResponse.success(component)
```

```
@componentAPI.get("/all_user")
async def read_user_components(current_user: User = Depends(get_current_user)):
    components = await
Component.filter(user=current_user).prefetch_related('model').values('id','name','location','updated_time','model__name','model__id','life_forecast','status')
    return CommonResponse.success(components)

@componentAPI.get("/model_count", description="返回当前用户下各模型的组件数量", tags=["charts"])
async def read_model_count(current_user: User = Depends(get_current_user)):
    component = await Component.filter(user=current_user)
    model_count = {}
    for c in component:
        if c.name in model_count:
            model_count[c.name] += 1
        else:
            model_count[c.name] = 1
    return CommonResponse.success(model_count)

@componentAPI.get("/location_count", description="返回当前用户下各位置的组件数量", tags=["charts"])
async def read_location_count(current_user: User = Depends(get_current_user)):
    component = await Component.filter(user=current_user)
    location_count = {}
    for c in component:
        if c.location in location_count:
            location_count[c.location] += 1
        else:
            location_count[c.location] = 1
    return CommonResponse.success(location_count)

@componentAPI.get("/pic/{id}")
async def read_component_pic(id: int, current_user: User = Depends(get_current_user)):
    component = await Component.get(id=id).prefetch_related('user')
    component_user = component.user
    if component.user != current_user:
        return CommonResponse.error(124, "无权访问其它用户的组件信息")
    return Response(content=component.pic, media_type="image/png")

@componentAPI.delete("/delete/{id}")
async def delete_component(id: int, current_user: User = Depends(get_current_user)):
    component = await Component.get(id=id).prefetch_related('user')
    component_user = component.user
    if component.user != current_user:
        return CommonResponse.error(124, "无权删除其它用户的组件信息")
    await component.delete()
    return CommonResponse.success("删除成功")


from fastapi import UploadFile, Form, Depends, Response
from common.CommonResponse import CommonResponse
from tools.security import get_current_user
from fastapi import APIRouter
from database.models import DData, Component
from common.CommonResponse import CommonResponse
from tortoise.exceptions import DoesNotExist
dataAPI = APIRouter()


@dataAPI.post("/", description="上传数据")
async def create_data(file: UploadFile, name: str = Form(...), component: int = Form(...), 
current_user=Depends(get_current_user)):
    """
    上传用户的指定组件的数据至数据库

    Args:
    - file: 上传的文件
    - name: 数据名称
    - component: 组件 id
    - current_user: 当前用户, 由 Depends(get_current_user)注入, 即需要在请求头中携带 token

    Returns:
    - 成功: 返回数据 id
    - 失败: 返回错误信息
    """
    try:
        component = await Component.get(id=component).prefetch_related('user')
    except DoesNotExist:
        return CommonResponse.error(104, "组件不存在")
    if not component:
        return CommonResponse.error(104, "组件不存在")
    if component.user != current_user:
        return CommonResponse.error(124, "无权访问其它用户的组件信息")
    data = await DData.create(
        file=await file.read(),
        name=name,
        component=component
    )
    return CommonResponse.success(data.id)


@dataAPI.get("/user_component/{id}", description="返回当前用户下的某个组件的数据",
response_model_exclude={"file"})
async def read_user_component_data(id: int, current_user=Depends(get_current_user)):
    component = await Component.get(id=id).prefetch_related('user')
    if not component:
        return CommonResponse.error(104, "组件不存在")
    if component.user != current_user:
        return CommonResponse.error(124, "无权访问其它用户的组件信息")
    datas = await DData.filter(component=component)
    for d in datas:
        d.file = ''
    return datas


@dataAPI.get("/download/{id}", description="下载数据")
async def download_data(id: int, current_user=Depends(get_current_user)):
    data = await DData.get(id=id).prefetch_related('component__user')
    if not data:
        return CommonResponse.error(105, "数据不存在")
    if data.component.user != current_user:
        return CommonResponse.error(125, "无权下载其它用户的数据")
    return Response(content=data.file, media_type="application/octet-stream", headers={"Content-Disposition":
f"attachment; filename={data.name}"})


@dataAPI.get("/all_user", description="返回当前用户下的所有数据", response_model_exclude={"file"})
async def read_user_data(current_user=Depends(get_current_user)):
    datas = await DData.filter(component__user=current_user).prefetch_related('component').values('id', 'name',
'time', 'result', 'component_id', 'component__name')
    return CommonResponse.success(datas)


@dataAPI.delete("/{id}", description="删除数据")
async def delete_data(id: int, current_user=Depends(get_current_user)):
    data = await DData.get(id=id).prefetch_related('component__user').values('id', 'component__user__username')
    if data['component__user__username'] != current_user.username:
        return CommonResponse.error(125, "无权删除其它用户的数据")
    delete_count = await DData.filter(id=id).delete()
    if delete_count == 0:
        return CommonResponse.error(500, "删除失败")
    return CommonResponse.success("删除成功")

from fastapi import APIRouter, Form, UploadFile, Depends, HTTPException, File
from database.models import MModel, User
from tools.security import get_current_user
from tools.md5 import get_file_md5
from common.CommonResponse import CommonResponse
import os

modelAPI = APIRouter()

@modelAPI.get("/all")
async def read_models():
    models = await MModel.all().values('id', 'name', 'style', 'status', 'description', 'uploaded_time', 'md5')
    # 这里使用 values 过滤掉了 modelfile 字段, 可以加快查询速度
    return CommonResponse.success(models)

@modelAPI.post("/", response_model_exclude={"modelfile"})
async def create_model(file: UploadFile, name: str = Form(...), style: str = Form(...),
                       status: str = Form(...), description: str = Form(...), user: User = Depends(get_current_user)):
    # 检查是否是合法的模型文件
    # TODO

    # 检查模型名是否重复
    file_content = await file.read()
    if await MModel.exists(name=name):
        return CommonResponse.error(100, "模型名已存在")
    md5 = await get_file_md5(file_content)
    if await MModel.exists(md5=md5):
        return CommonResponse.error(101, "模型文件已存在")
    model = await MModel.create(name=name, style=style, status=status, description=description,
modelfile=file_content, md5=md5,user=user)
    # 将模型文件存储到本地
    # 如果目录不存在则创建
    if not os.path.exists("./model_files"):
        os.makedirs("./model_files")
    with open(f"./model_files/{md5}", "wb") as f:
        f.write(file_content)
    return CommonResponse.success(model)

@modelAPI.get("/style", description="获取各模型风格占百分比", tags=["charts"])
async def style():
    style = await MModel.all().values_list('style', flat=True)
    total = len(style)
    style_dict = {}
    for s in style:
        if s in style_dict:
            style_dict[s] += 1
        else:
            style_dict[s] = 1
    for k in style_dict:
        style_dict[k] /= total
    return CommonResponse.success(style_dict)

@modelAPI.get("/status", description="获取各模型状态占百分比", tags=["charts"])
async def status():
    status = await MModel.all().values_list('status', flat=True)
    total = len(status)
    status_dict = {}
    for s in status:
        if s in status_dict:
            status_dict[s] += 1
        else:
            status_dict[s] = 1
    for k in status_dict:
        status_dict[k] /= total
    return CommonResponse.success(status_dict)

# 获取指定 id 的模型

@modelAPI.get("/{id}")
async def read_model(id: int):
    model = await MModel.get(id=id)
    # 判断模型文件是否存在于目录
    if not os.path.exists("./model_files"):
        os.makedirs("D:/model_files")
    if not os.path.exists(f"./model_files/{model.md5}"):
        with open(f"./model_files/{model.md5}", "wb") as f:
            f.write(model.modelfile)

    if not model:
        return CommonResponse.error(103, "模型不存在")
    return CommonResponse.success(model)

@modelAPI.delete("/{id}")
async def delete_model(id: int, user: User = Depends(get_current_user)):
    model = await MModel.get(id=id).prefetch_related('user').values('id', 'user__username')
    if model['user__username'] is not None:
        if model['user__username'] != user.username:
            return CommonResponse.error(123, "无权删除其它用户的模型")
    deleted_count = await MModel.filter(id=id).delete()
    if deleted_count == 0:
        return CommonResponse.error(500, "删除失败")
    return CommonResponse.success("删除成功!")

from fastapi import APIRouter, Depends, HTTPException, Form, status,UploadFile,Response
from database.models import User
from fastapi.security import OAuth2PasswordRequestForm
from tools.security import get_password_hash, create_access_token, authenticate_user, get_current_user, Token
from common.CommonResponse import CommonResponse
userAPI = APIRouter()

@userAPI.get("/current_user")
async def read_users_me(current_user: User = Depends(get_current_user)):
    return current_user

@userAPI.post("/token")
async def login_token(form_data: OAuth2PasswordRequestForm = Depends()) -> Token:
    user = await authenticate_user(form_data.username, form_data.password)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Incorrect username or password",
            headers={"WWW-Authenticate": "Bearer"},
        )
    return Token(access_token=await create_access_token(data={"sub": user.username}), token_type="Bearer")
```

```
@userAPI.post("/register")
async def register(avatar:UploadFile,username: str = Form(...), password: str = Form(...), email: str = Form(...),
phone: str = Form(...)):
    if await User.exists(username=username):
        return CommonResponse.error(120, "用户名已存在")
    hashed_password = get_password_hash(password)
    user = await User.create(username=username, hashed_password=hashed_password, email=email,
phone=phone, avatar=await avatar.read())
    return CommonResponse.success(Token(access_token=await create_access_token(data={"sub":
user.username}), token_type="Bearer").to_json())



@userAPI.post("/login")
async def login(form_data: OAuth2PasswordRequestForm = Depends()) -> Token:
    user = await authenticate_user(form_data.username, form_data.password)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Incorrect username or password",
            headers={"WWW-Authenticate": "Bearer"},
        )
    return CommonResponse.success(Token(access_token=await create_access_token(data={"sub":
user.username}), token_type="Bearer").to_json())


@userAPI.get("/avatar")
async def get_avatar(current_user=Depends(get_current_user)):
    return Response(content=current_user.avatar, media_type="image/png")
from fastapi import status
from fastapi.responses import JSONResponse, Response
from typing import Union
import json
from json import JSONEncoder as JSONR
from database.models import User, MModel, Component, DData
import datetime


class CommonResponse:
    @staticmethod
    def success(data, message: str = "success") -> JSONResponse:
        cls = Myncoder
        content = json.dumps({
            "code": 200,
            "message": message,
            "data": data
        }, cls=cls, ensure_ascii=False)
        return Response(content=content, status_code=status.HTTP_200_OK, media_type="application/json")

    @staticmethod
    def error(code: int, message: str) -> JSONResponse:
        return JSONResponse(
            status_code=status.HTTP_200_OK,
            content={
                "code": code,
                "message": message,
                "data": None
            }
        )

    @staticmethod
    def custom(status_code: int, code: int, message: str, data: Union[dict, list, str] = None) -> JSONResponse:
        return JSONResponse(
            status_code=status_code,
            content={
                "code": code,
                "message": message,
                "data": data
            }
        )

    @staticmethod
    def response(status_code: int, code: int, message: str, data: Union[dict, list, str] = None) -> JSONResponse:
        return JSONResponse(
            status_code=status_code,
            content={
                "code": code,
                "message": message,
                "data": data
            }
        )

class Myncoder(JSONR):
    def default(self, o):
        if isinstance(o, Component):
            return {
                "id": o.id,
                "name": o.name,
                "status": o.status,
                "life_forecast": o.life_forecast,
                "location": o.location,
                "updated_time": o.updated_time.strftime('%Y-%m-%d %H:%M:%S'),
                "model_name": o.model.name if isinstance(o.model, MModel) else None,
                "model_id": o.model.id if isinstance(o.model, MModel) else None,
            }
        if isinstance(o, DData):
            return {
                "id": o.id,
                "name": o.name,
                "time": o.time,
                "result": o.result,
                "component_name": o.component.name if isinstance(o.component, Component) else None,
                "component_id": o.component.id if isinstance(o.component, Component) else None
            }
        if isinstance(o, MModel):
            return {
                "id": o.id,
                "name": o.name,
                "style": o.style,
                "uploaded_time": o.uploaded_time.strftime('%Y-%m-%d %H:%M:%S'),
                "status": o.status,
                "description": o.description,
                "md5": o.md5,
                "user": o.user.username if o.user else None
            }
        if isinstance(o, datetime.datetime):
            # 转化成东八区时间
            o = o + datetime.timedelta(hours=8)
            return o.strftime('%Y-%m-%d %H:%M:%S')
        if isinstance(o, list):
            return [self.default(i) for i in o]
        if isinstance(o, dict):
            return {k: self.default(v) for k, v in o.items()}
        return super().default(o)
```

```
CONFIG = {
    "connections": {
        "default": {
            "engine": "tortoise.backends.mysql",  # 数据库类型
            "credentials": {
                "host": "111.173.119.228",
                "port": "14082",
                "user": "root",
                "password": "18937082731",
                "database": "testdb"
            },
        },
    },
    "apps": {
        "models": {
            "models": ["database.models", "aerich.models"],
            "default_connection": "default",
        }
    },
}

from tortoise.models import Model
from tortoise import fields

class User(Model):
    username = fields.CharField(pk=True, max_length=50, description="用户名")
    hashed_password = fields.CharField(max_length=128, description="密码")
    email = fields.CharField(max_length=50, description="邮箱")
    phone = fields.CharField(max_length=50, description="电话")
    role = fields.IntField(description="角色", default=0)
    avatar = fields.BinaryField(description="头像", null=True)
    # 与 model 的一对多关系
    models: fields.ReverseRelation['MModel']

class MModel(Model):
    id = fields.IntField(pk=True)
    name = fields.CharField(max_length=50, description="模型名称", unique=True)
    style = fields.CharField(
        max_length=50, description="模型风格", default="default")
    uploaded_time = fields.DatetimeField(auto_now_add=True, description="上传时间")
    status = fields.CharField(
        max_length=10, description="模型状态", default="未知")
    description = fields.TextField(description="描述")
    modelfile = fields.BinaryField(description="模型文件")
    md5 = fields.CharField(max_length=32, description="md5 值", unique=True)
    user = fields.ForeignKeyField(
        'models.User', related_name='models', description="用户", null=True)
    # 与 component 的一对多关系
    components: fields.ReverseRelation['Component']

class Component(Model):
    id = fields.IntField(pk=True)
    name = fields.CharField(max_length=50, description="组件名称")
    status = fields.IntField(description="组件状态", null=True)
    life_forecast = fields.IntField(description="寿命预测", default=-1)
    location = fields.CharField(max_length=50, description="位置")
    updated_time = fields.DatetimeField(auto_now_add=True, description="更新时间")
    pic = fields.BinaryField(description="图片", null=True)
    model = fields.ForeignKeyField(
        'models.MModel', related_name='components', description="模型", null=True)
    user = fields.ForeignKeyField(
        'models.User', related_name='components', description="用户")
    description = fields.TextField(description="描述", null=True)
    # 与 data 的一对多关系
    data: fields.ReverseRelation['DData']

class DData(Model):
    id = fields.IntField(pk=True)
    file = fields.BinaryField(description="数据文件")
    name = fields.CharField(max_length=50, description="数据名称", null=True)
    time = fields.DatetimeField(auto_now_add=True, description="上传时间")
    result = fields.TextField(description="结果", null=True)
    component = fields.ForeignKeyField(
        'models.Component', related_name='data', description="组件")

import hashlib
from fastapi import UploadFile
```

```
async def get_file_md5(file: bytes):
    md5 = hashlib.md5()
    md5.update(file)
    return md5.hexdigest()


import bcrypt
from datetime import datetime, timedelta, timezone
from typing import Union
from passlib.context import CryptContext
from typing import Union
from database.models import User
from jose import ExpiredSignatureError, JWTError, jwt
from fastapi.security import OAuth2PasswordBearer
from pydantic import BaseModel
from fastapi import Depends, HTTPException, status
SECRET_KEY =
"7e93a0a11669ba8adeb21097daf302569bad252992cc1c2bf929f606b2be23f045575c790b18f1ba41be715a6a2ce
833c894d78da1a40ea7ac44b5cbefd98e25"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30


pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="user/token")


class Token(BaseModel):
    access_token: str
    token_type: str


    def to_json(self):
        return {
            "token": self.access_token,
            "type": self.token_type
        }


    def to_str(self):
        return f"token: {self.access_token}, type: {self.token_type}"


class TokenData(BaseModel):
    username: Union[str, None] = None
```

```
def get_oauth2_scheme():
    return oauth2_scheme

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)

def get_password_hash(password):
    return pwd_context.hash(password)

async def authenticate_user(username: str, password: str):
    try:
        user = await User.get(username=username)
    except:
        return False
    if not verify_password(password, user.hashed_password):
        return False
    return user

async def create_access_token(data: dict, expires_delta: Union[timedelta, None] = None):
    to_encode = data.copy()
    if expires_delta:
        expire = datetime.utcnow() + expires_delta
    else:
        expire = datetime.utcnow() + timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
    return encoded_jwt

async def get_current_user(token: str = Depends(oauth2_scheme)):
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        if username is None:
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Could not validate credentials"
            )
        token_data = TokenData(username=username)
    except ExpiredSignatureError:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Token 已过期，请重新登录"
        )
    except JWTError:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Could not validate credentials"
        )
    user = await User.get(username=token_data.username)
    if user is None:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )
    return user

async def is_expired(token: str) -> str:
    try:
        new_token = None
        payload = jwt.decode(token, SECRET_KEY, algorithms=[
                          ALGORITHM], options={"verify_exp": False})
        exp = payload.get("exp")
        username = payload.get("sub")
        if exp is not None:
            # 如果过期时间是前一分钟, 则产生新的 token
            if exp - datetime.now(timezone.utc) < timedelta(minutes=1):
                new_token = await create_access_token(data={"sub": username})
        return new_token
    except Exception as e:
        return None

import subprocess
from fastapi import FastAPI, File, Form, Request, status, Depends
from jose import ExpiredSignatureError
from common.CommonResponse import CommonResponse
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from tortoise.exceptions import DoesNotExist
from tortoise.contrib.fastapi import register_tortoise
from database.config import CONFIG
from database.models import User
from tools.security import get_current_user, is_expired
from starlette.exceptions import HTTPException as StarletteHTTPException

from api.user import userAPI
from api.mmodel import modelAPI
from api.component import componentAPI
from api.data import dataAPI

uploaded_files_md5s = {}

class Item(BaseModel):
    data: str

class PieData(BaseModel):
    value: float
    name: str

app = FastAPI()
register_tortoise(app=app,
                  config=CONFIG,
                  generate_schemas=True,  # 是否自动生成表结构
                  )
app.include_router(userAPI, prefix="/user", tags=["用户相关 API"])
app.include_router(modelAPI, prefix="/model", tags=["模型相关 API"])
app.include_router(componentAPI, prefix="/component", tags=["组件相关 API"])
app.include_router(dataAPI, prefix="/data", tags=["数据相关 API"])

@app.middleware("http")
async def token_expired_generate(request: Request, call_next):
    # 如果 token 过期, 产生新的 token
    bearer_token = request.headers.get("Authorization")
    response = await call_next(request)
    if bearer_token:
        token = bearer_token.split(" ")[1]
        new_token = is_expired(token)
        if new_token:
            response.set_cookie("Authorization", new_token)
            return response
    return response



@app.exception_handler(StarletteHTTPException)
async def http_exception_handler(request, exc):
    # 处理 HTTPException 异常
    return CommonResponse.error(exc.status_code, exc.detail)

@app.exception_handler(DoesNotExist)
async def does_not_exist_exception_handler(request, exc):
    return CommonResponse.error(404, "资源不存在")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)



@app.get("/current_user")
async def read_users_me(current_user: User = Depends(get_current_user)):
    return current_user

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app="main:app", host="localhost", port=8000, reload=True)
```