```
@Entry
@Component
export struct Index {
  @State token: string = ''
  @State renter: string = 'demo'
  @State account: string = 'admin'
  @State password: string = 'mael@008!@#'
  @State url: string = ''
  onPageShow(): void {
    window.getLastWindow(getContext(this)).then((windowStage: window.Window)
=> {
      windowStage.setWindowPrivacyMode(true);
    });
  }
  onPageHide(): void {
    window.getLastWindow(getContext(this)).then((windowStage: window.Window)
=> {
      windowStage.setWindowPrivacyMode(false);
    });
  }
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  httpRequest(url: string, params?: LoginParam): Promise<ResponseResult> {
    let httpRequest = http.createHttp();
    let responseResult = httpRequest.request(url, {
      method: http.RequestMethod.POST,
      header: {
        'Content-Type': 'application/json',
        'Authorization': 'Basic c2FiZXI6c2FiZXJfc2VjcmV0',
        'Tenant-Id': this.renter
      },
      extraData: params
    });
    let serverData: ResponseResult = new ResponseResult();
    return responseResult.then((value: http.HttpResponse) => {
      if (value.responseCode === 200) {
        let result = `${value.result}`;
        let resultJson: ResponseResult = JSON.parse(result);
        serverData.access_token = resultJson.access_token;
        serverData.user_name = resultJson.user_name;
        serverData.avatar = resultJson.avatar;
        serverData.user_id = resultJson.user_id
      } else {
        serverData.msg =
`${$r('app.string.http_error_message')}&${value.responseCode}`;
      }
      return serverData;
    }).catch(() => {
      serverData.msg = $r('app.string.http_error_message');
      return serverData;
    })
  }
  async getPreference(key: string) {
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    return (await preference.get(key, '')).toString();
  }
  async putPreference(key: string, value: string) {
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    await preference.put(key, value);
```

```
    await preference.flush();
  }
  async md5(password: string) {
    let mdAlgName = 'MD5'; // 使用 MD5 算法
    let message = password; // 需要加密的密码
    let md = await cryptoFramework.createMd(mdAlgName); // 创建 MD5 摘要实例
    // 将字符串转换为 Uint8Array，并进行摘要计算
    await md.update({ data: new Uint8Array(buffer.from(message,
'utf-8').buffer) });
    let mdResult = await md.digest(); // 获取摘要结果
    let hexString = Array.from(mdResult.data)
      .map(byte => byte.toString(16).padStart(2, '0'))
      .join('');
    return hexString;
  }
  async aboutToAppear() {
    this.url = await this.getPreference('url')
    console.log('url', this.url)
    if (!this.url) {
      if (dataPreferences) {
        this.url = dataPreferences.getSync('url', Constants.Default_SERVER) as
string;
      }
    }
  }
  build() {
    Column() {
      Image($r('app.media.login_back')).width('100%')
      Column() {
        Row() {
          Image($r('app.media.renter')).width(30)
          TextInput({ placeholder: '请输入租户编号', text: this.renter })
            .type(InputType.Normal)
            .margin({
              left: 20
            })
            .onChange((value: string) => {
              this.renter = value
            })
        }
        .margin({
          top: 10
        })
        Column() {
          Row() {
            Image($r('app.media.account')).width(30)
            TextInput({ placeholder: '请输入用户名', text: this.account })
              .margin({
                left: 20
              })
              .type(InputType.Normal)
              .onChange((value: string) => {
                this.account = value
              })
          }.margin({
            top: 10
          })
        }
        Column() {
          Row() {
            Image($r('app.media.password')).width(30)
            TextInput({ placeholder: '请输入密码', text: this.password })
              .margin({
                left: 20
```

```
            })
            .type(InputType.Password)
            .onChange((value: string) => {
              this.password = value
            })
          }
          .margin({
            top: 10
          })
        }
        Row() {
          Button('登录')
            .onClick(() => {
              AlertDialog.show(
                {
                  title: '登录',
                  message: '是否登录？',
                  autoCancel: true,
                  alignment: DialogAlignment.Bottom,
                  gridCount: 4,
                  offset: { dx: 0, dy: -20 },
                  primaryButton: {
                    value: '取消',
                    action: () => {
                    }
                  },
                  secondaryButton: {
                    enabled: true,
                    defaultFocus: true,
                    style: DialogButtonStyle.HIGHLIGHT,
                    value: '确定',
                    action: async () => {
                      let password = await this.md5(this.password)
                      let loginParam:LoginParam = {
                        username: this.account,
                        password,
                        grant_type: 'password',
                        scope: 'all',
                        type: 'account'
                      }
                      this.httpRequest(this.url +
`/api/blade-auth/oauth/token?username=${loginParam.username}&password=${log
inParam.password}&grant_type=${loginParam.grant_type}&scope=${loginParam.sc
ope}&type=${loginParam.type}`).then((data: ResponseResult) => {
                        this.token = data.access_token
                        this.putPreference('token', this.token)
                        this.putPreference('username', data.user_name)
                        this.putPreference('avatar', data.avatar)
                        this.putPreference('userid', data.user_id)
                        router.pushUrl({
                          url: 'pages/Content'
                        })
                      })
                    }
                  }
                }
              )
            })
          Button('设置')
            .onClick(() => {
              router.pushUrl({
                url: 'pages/Setting'
              });
            })
```

```
          .margin({
            left: 10
          })
        }
        .margin({
          top: 10
        })
      }.width('90%')
      .padding(30)
    }
  }
}
@Entry
@Component
struct Content {
  private tabsController : TabsController = new TabsController();
  scroller: Scroller = new Scroller()
  @State currentIndex: number = 0;
  @State warnModel: WarnModel = new WarnModel()
  @State countList:Count[]=[]
  async onPageShow() {
    if(this.currentIndex==1){
      this.scroller.scrollTo({ xOffset: 0, yOffset: 0 })
      this.warnModel.currentPage = 1
      this.warnModel.hasMore = true
      this.changeWarning()
    }else if(this.currentIndex==3){
      this.countList=await this.getNotice()
    }
  }
  getWarning(queryData: SafeQueryData): Promise<QueryModel<WarnData>> {
     return new Promise(async (resolve: Function, reject: Function) => {
    httpRequestPost(Constants.SERVER+'/ml-mlink/equipmentalarm/page',
queryData)
      .then((result: ResponseResult) => {
        if (result && result.code === 200) {
                resolve(result.data);
        } else {
          reject('wrong');
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
  getNotice():Promise<Count[]>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        '/ml-mlink/tasknotice/groupTypeSum')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  changeWarning() {
    let queryData: SafeQueryData = new SafeQueryData();
```

```
    queryData.current = 1;
    queryData.size = 20;
    queryData.isRealTime = 1;
    queryData.alarmContent = ''
    this.getWarning(queryData).then((data) => {
      this.warnModel.pageState = PageState.Success;
        if (data.records.length === this.warnModel.size) {
        this.warnModel.currentPage++;
        this.warnModel.hasMore = true;
      } else {
        this.warnModel.hasMore = false;
      }
      this.warnModel.warnData = data.records
    });
  }
  @Builder
  tabBarBuilder(title: string, targetIndex: number, selectedIcon: Resource,
unselectIcon: Resource) {
    Column() {
      Image(this.currentIndex === targetIndex ? selectedIcon : unselectIcon)
        .width(24)
        .height(24)
      Text(title)
        .fontFamily('HarmonyHeiTi-Medium')
        .fontSize(10)
        .fontColor(this.currentIndex === targetIndex ? '#0A59F7' :
'rgba(0,0,0,0.60)')
        .textAlign(TextAlign.Center)
        .lineHeight(14)
        .fontWeight(500)
    }
    .width('100%')
    .height('100%')
    .justifyContent(FlexAlign.Center)
    .alignItems(HorizontalAlign.Center)
    .onClick(() => {
      this.currentIndex = targetIndex;
      this.tabsController.changeIndex(targetIndex);
    })
  }
  build() {
    Tabs({ barPosition: BarPosition.End,controller: this.tabsController  }){
      TabContent() {
        Work()
      }
      .tabBar(this.tabBarBuilder('工作中心', 0, $r('app.media.work_active'),
$r('app.media.work')))
      TabContent() {

Safe({changeWarning:this.changeWarning,getWarning:this.getWarning,warnModel:
this.warnModel,scroller:this.scroller})}
      .tabBar(this.tabBarBuilder('安全中心', 1, $r('app.media.safe_active'),
$r('app.media.safe')))
      TabContent() {
        Home()}
      .tabBar(this.tabBarBuilder('首页', 2, $r('app.media.home_active'),
$r('app.media.home')))
      TabContent() {
        Message({getNotice:this.getNotice,countList:this.countList})}
      .tabBar(this.tabBarBuilder('消息', 3, $r('app.media.message_active'),
$r('app.media.message')))
      TabContent() {
        Mine()}
      .tabBar(this.tabBarBuilder('我的', 4, $r('app.media.my_active'),
```

```
$r('app.media.my')))
    }
    .scrollable(false)
    .vertical(false)
    .divider({
      strokeWidth: 0.5,
      color: '#0D182431'
    })
    .backgroundColor('#F1F3F5')
  }
}
@Component
export default struct Home {
  @State web:string=''
  @State token:string = ''
  webController: webview.WebviewController = new webview.WebviewController();
  async aboutToAppear() {
    this.token = await this.getPreference();
    this.web =
Constants.Default_SERVER+`/mlink/grapheditor/dist/viewer_mp.html?token=${th
is.token}&authorization=c2FiZXI6c2FiZXJfc2VjcmV0&notabbar=1`
    this.webController.loadUrl(this.web);
  }
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  async getPreference() {
    let token = '';
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    token = (await preference.get('token', '')).toString();
    return token;
  }
  build() {
    Column(){
      Web({ src:  '', controller: this.webController })
    }
  }
}
@Component
export default struct Work {
  @State assignedPersonId: string = ''
  @State inspect: number = 0
  @State maintain: number = 0
  @State repair: number = 0
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  async getPreference(key: string) {
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    return (await preference.get(key, '')).toString();
  }
  getTaskCount(id: string): Promise<TaskCount[]> {
    return new Promise(async (resolve: Function, reject: Function) => {
httpRequestGet(Constants.SERVER+'/ml-mlink/repairtask/groupTypeSum?assigned
PersonId=' + id)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
```

```
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
async aboutToAppear() {
  this.assignedPersonId = await this.getPreference('userid');
  this.getTaskCount(this.assignedPersonId).then(res => {
    this.inspect = res.find((item) => item.name === '巡检')?.value ?? 0;
    this.maintain = res.find((item) => item.name === '保养')?.value ?? 0;
    this.repair = res.find((item) => item.name === '维修')?.value ?? 0;
  })
}
build() {
    Scroll() {
      Column() {
        Column() {
          Row() {
            Text('我的任务')
          }
          .width('100%')
          .justifyContent(FlexAlign.Start)
          .height('20%')
          Row() {
            Column() {
              Text(String(this.inspect))
              Text('巡检')
            }
            Column() {
              Text(String(this.maintain))
              Text('保养')
            }
            Column() {
              Text(String(this.repair.toString()))
              Text('维修')
            }
          }
          .height('80%')
          .width('100%')
          .justifyContent(FlexAlign.SpaceAround)
        }
        .padding({
          top: 20,
          left: 30,
          right: 30
        })
        .borderRadius(30)
        .height(150)
        .width('100%')
        .linearGradient({
          angle: 180,
          colors: [['rgba(137, 199, 255, 1)', 0.0], ['rgba(137, 199, 255, 0.5)',
1.0]]
        })
        Column() {
          Text('提交工单')
            .margin({
              left: 30
            })
        }
        .height(60)
        .borderRadius(30)
```

```
        .width('100%')
        .backgroundColor(0xffffff)
        .justifyContent(FlexAlign.Center)
        .alignItems(HorizontalAlign.Start)
        .margin({
          top: 10
        })
        Column() {
          Row() {
            Row() {
              Column() {
                Image($r('app.media.report')).width(40)
              }
              .justifyContent(FlexAlign.Center)
              .backgroundColor(0x0081ff)
              .borderRadius(50)
              .width(50)
              .height(50)
              Text('提交报修').fontColor(0x2d89e6).margin({
                left: 10
              })
            }
            Image($r('app.media.repair')).width(80)
          }
          .justifyContent(FlexAlign.SpaceBetween)
          .alignItems(VerticalAlign.Center)
          .width('100%')
          .height('100%')
          .padding({
            left: 20,
            right: 20
          })
        }
        .onClick(()=>{
          router.pushUrl({
            url: 'pages/RepairInfo'
          });
        })
        .height(100)
        .borderRadius(30)
        .backgroundColor(0xafd6fa)
        .width('100%')
        .margin({
          top: 10
        })
        Column() {
          Text('工作查询')
            .margin({
              left: 30
            })
        }
        .height(60)
        .borderRadius(30)
        .width('100%')
        .backgroundColor(0xffffff)
        .justifyContent(FlexAlign.Center)
        .alignItems(HorizontalAlign.Start)
        .margin({
          top: 10
        })
        Column() {
          Grid() {
            GridItem() {
              Row() {
```

```
        Column() {
          Image($r('app.media.address')).width(30)
        }
        .justifyContent(FlexAlign.Center)
        .backgroundColor(0x0081ff)
        .borderRadius(50)
        .width(50)
        .height(50)
        Text('巡检查询').margin({
          left: 10
        })
      }
      .backgroundColor(0xffffff)
      .borderRadius(15)
      .width('100%')
      .height(100)
      .padding(20)
    }
    .onClick(() => {
      router.pushUrl({
        url: 'pages/Inspect'
      });
    })
    GridItem() {
      Row() {
        Column() {
          Image($r('app.media.keep')).width(30)
        }
        .justifyContent(FlexAlign.Center)
        .backgroundColor(0x13df7c)
        .borderRadius(50)
        .width(50)
        .height(50)
        Text('保养查询')
          .margin({
            left: 10
          })
      }
      .backgroundColor(0xffffff)
      .borderRadius(15)
      .width('100%')
      .height(100)
      .padding(20)
    }
    .onClick(() => {
                  router.pushUrl({
        url: 'pages/Maintenance'
      });
    })
    GridItem() {
      Row() {
        Column() {
          Image($r('app.media.fix')).width(30)
        }
        .justifyContent(FlexAlign.Center)
        .backgroundColor(0xfd5635)
        .borderRadius(50)
        .width(50)
        .height(50)
        Text('维修查询')
          .margin({
            left: 10
          })
      }
```

```
              .backgroundColor(0xffffff)
              .borderRadius(15)
              .width('100%')
              .height(100)
              .padding(20)
            }
            .onClick(() => {
              router.pushUrl({
                url: 'pages/Repair',
                params:{
                  type:'repair'
                }
              });
            })
            GridItem() {
              Row() {
                Column() {
                  Image($r('app.media.address')).width(30)
                }
                .justifyContent(FlexAlign.Center)
                .backgroundColor(0xf5cc39)
                .borderRadius(50)
                .width(50)
                .height(50)
                Text('报修查询').margin({
                  left: 10
                })
              }
              .backgroundColor(0xffffff)
              .borderRadius(15)
              .width('100%')
              .height(100)
              .padding(20)
            }
            .onClick(() => {
              router.pushUrl({
                url: 'pages/Repair',
                params:{
                  type:'report'
                }
              });
            })
          }
          .margin({
            top: 10
          })
          .rowsTemplate('1fr 1fr')
          .columnsTemplate('1fr 1fr')
          .height(215)
          .columnsGap(10)
          .rowsGap(10)
        }
      }
      .padding({
        top:20,
        left:10,
        right:10
      })
    }.align(Alignment.Top).height('100%')
  }
}
@Entry
@Component
struct Inspect {
```

```
  @State fontColor: string = '#182431'
  @State selectedFontColor: string = '#007DFF'
  @State currentIndex: number = 0
  @State selectedIndex: number = 0
  @State inspectList1: Info[] = []
  @State inspectList2: Info[] = []
  @State inspectList3: Info[] = []
  hasMore1:boolean = false
  hasMore2:boolean = false
  hasMore3:boolean = false
  private controller: TabsController = new TabsController()
  pageSize: number = 20
  page1: number = 1
  page2: number = 1
  page3: number = 1
  async aboutToAppear() {
  }
  async onPageShow() {
    this.page1=1
    this.page2=1
    this.page3=1
    let queryParameter1: QueryData = {
      size: this.pageSize,
      current: this.page1,
      taskStatus: 0
    }
    let queryParameter2: QueryData = {
      size: this.pageSize,
      current: this.page2,
      taskStatus: 1
    }
    let queryParameter3: QueryData = {
      size: this.pageSize,
      current: this.page3,
      taskStatus: 3
    }
    let data1 = await this.getInspectList(queryParameter1)
    let data2 = await this.getInspectList(queryParameter2)
    let data3 = await this.getInspectList(queryParameter3)
    this.inspectList1 = data1.records
    if(this.inspectList1.length==data1.size){
      this.page1++
      this.hasMore1 = true
    }
    this.inspectList2 = data2.records
    if(this.inspectList2.length==data2.size){
      this.page2++
      this.hasMore2 = true
    }
    this.inspectList3 = data3.records
    if(this.inspectList3.length==data3.size){
      this.page3++
      this.hasMore3 = true
    }
  }
  getInspectList(queryData: QueryData): Promise<QueryModel<Info>> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +

`/ml-mlink/inspectiontask/page?size=${queryData.size}&current=${queryData.c
urrent}&taskStatus=${queryData.taskStatus}`)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
```

```
        } else {
          reject('wrong');
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
updateInspectList(queryData:QueryData){
  if(this.currentIndex==0){
    if(this.hasMore1){
      this.getInspectList(queryData).then((result)=> {
        let data = result.records
        if(data.length==result.size){
          this.page1++
          this.hasMore1 = true
        }else {
          this.hasMore1 = false
        }
        this.inspectList1 = this.inspectList1.concat(data)
      })
    }else {
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
              console.info('Button-clicking callback')
            }
          },
          cancel: () => {
            console.info('Closed callbacks')
          },
        })
    }
  }else if(this.currentIndex==1){
    if(this.hasMore2){
      this.getInspectList(queryData).then((result)=> {
        let data = result.records
        if(data.length==result.size){
          this.page2++
          this.hasMore2 = true
        }else {
          this.hasMore2 = false
        }
        this.inspectList2 = this.inspectList2.concat(data)
      })
    }else {
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
              console.info('Button-clicking callback')
            }
```

```
            },
            cancel: () => {
              console.info('Closed callbacks')
            },
          })
      }
    }else {
      if(this.hasMore3){
        this.getInspectList(queryData).then((result)=> {
          let data = result.records
          if(data.length==result.size){
            this.page3++
            this.hasMore3 = true
          }else {
            this.hasMore3 = false
          }
          this.inspectList3 = this.inspectList3.concat(data)
        })
      } else {
        AlertDialog.show(
          {
            title: '提示',
            message: '已没有更多数据',
            autoCancel: true,
            alignment: DialogAlignment.Center,
            confirm: {
              value: '确定',
              action: () => {
                console.info('Button-clicking callback')
              }
            },
            cancel: () => {
              console.info('Closed callbacks')
            },
          })
      }
    }
  }
  @Builder
  tabBuilder(index: number, name: string) {
    Column() {
      Text(name)
        .fontColor(this.selectedIndex === index ? this.selectedFontColor :
this.fontColor)
        .fontSize(16)
        .fontWeight(this.selectedIndex === index ? 500 : 400)
        .lineHeight(22)
        .margin({ top: 17, bottom: 7 })
      Divider()
        .strokeWidth(2)
        .color('#007DFF')
        .opacity(this.selectedIndex === index ? 1 : 0)
    }.width('100%')
  }
  @Builder
  inspectList(params: param) {
    Scroll() {
      Column() {
        ForEach(params.list, (item: Info) => {
          Row() {
            if (item.taskStatus == 0) {
              Column() {
                Text('未检').fontColor(0xFFFFFF)
              }.backgroundColor(0xFF6766).width(50).height('100%').justifyCo
```

```
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            } else if (item.taskStatus == 1) {
              Column() {
                Text('进行中').fontColor(0xFFFFFF)
              }.backgroundColor(0x47E37E).width(50).height('100%').justifyCo
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            } else {
              Column() {
                Text('已完成').fontColor(0xFFFFFF)
              }.backgroundColor(0x4B93E7).width(50).height('100%').justifyCo
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            }
            Column() {
              Text(item.createTime)
              Text(item.planName).margin({
                top:15
              })
            }.layoutWeight(1).alignItems(HorizontalAlign.Start).margin({
              left:10,
            })
            Image($r('app.media.arrow')).width(20)
          }.height(100)
          .justifyContent(FlexAlign.Center)
          .margin({
            top:10,
            left:10,
            right:10
          })
          .padding({
            right:15
          })
          .onClick(() => {
            router.pushUrl({
              url: 'pages/InspectDetail',
              params:{
                inspectionPlanId:item.inspectionPlanId,
                taskId:item.id
              }
            });
          })
```

```
          .alignItems(VerticalAlign.Center)
          .backgroundColor(0xFFFFFF)
          .borderRadius(20)
          Divider().strokeWidth(3).color('#F1F3F5')
        }, (item: Info) => item.id)
      }.width('100%')
    }.layoutWeight(1).align(Alignment.Top).height('100%').onScrollEdge((sid
e: Edge) => {
      if (side == Edge.Bottom) {
        if(this.currentIndex==0){
          let queryParameter1: QueryData = {
            size: this.pageSize,
            current: this.page1,
            taskStatus: 0
          }
            this.updateInspectList(queryParameter1)
        }else if(this.currentIndex==1){
          let queryParameter2: QueryData = {
            size: this.pageSize,
            current: this.page2,
            taskStatus: 1
          }
          this.updateInspectList(queryParameter2)
        }else {
          let queryParameter3: QueryData = {
            size: this.pageSize,
            current: this.page3,
            taskStatus: 3
          }
          this.updateInspectList(queryParameter3)
        }
      }
    })
  }
  build() {
    Column() {
      Tabs({ barPosition: BarPosition.Start, index: this.currentIndex,
controller: this.controller }) {
        TabContent() {
          this.inspectList({ list: this.inspectList1 })
        }.tabBar(this.tabBuilder(0, '未检'))
        TabContent() {
          this.inspectList({ list: this.inspectList2 })
        }.tabBar(this.tabBuilder(1, '进行中'))
        TabContent() {
          this.inspectList({ list: this.inspectList3 })
        }.tabBar(this.tabBuilder(2, '已完成'))
      }
      .vertical(false)
      .barMode(BarMode.Fixed)
      .barWidth(360)
      .barHeight(56)
      .animationDuration(400)
      .onChange((index: number) => {
        this.currentIndex = index
      })
      .onAnimationStart((index: number, targetIndex: number, event:
TabsAnimationEvent) => {
        if (index === targetIndex) {
          return
        }
        // selectedIndex 控制自定义 TabBar 内 Image 和 Text 颜色切换
        this.selectedIndex = targetIndex
      })
```

```
      .width('100%')
      .height('100%')
      .backgroundColor('#F1F3F5')
    }.width('100%')
  }
}
@Entry
@Component
struct InspectDetail {
  opsInspectionNormPicVO:Array<Norm>=[]
  @State imgObject:ImgObject = []
  @State message: string = 'InspectDetail';
  @State inspectionPlanId:string = ''
  @State taskId:string = ''
  @State taskStatus:string=''
  @State status:number = 0
  @State btnTxt:string=''
  @State stationList:Station[]=[{name:'',id:''}]
  @State picList:string[]=[];
  @State imageUri: string = '';
  @State stationName:string | undefined = ''
  selectedDate: Date =  new Date();
  remarkList:string[]=[]
  inspectionMethodOptions:Option[]=  [{
    label: '目视',
    value: 1
  },
    {
      label: '目测',
      value: 2
    },
    {
      label: '目视 & 测量',
      value: 3
    },
    {
      label: '目视 & 手动',
      value: 4
    },
    {
      label: '目视 & 测温仪',
      value: 5
    },
    {
      label: '目视 & 听 & 闻 & 手动 & 测量',
      value: 6
    }
  ]
  taskStatusOptions:Option[]= [{
    label: '未检',
    value: 0
  },
    {
      label: '进行中',
      value: 1
    },
    {
      label: '暂停',
      value: 2
    },
    {
      label: '已检',
      value: 3
    },
```

```
    {
      label: '已取消',
      value: 4
    }
  ]
  @State inspectStandardList:InspectStandard[] = [{
    inspectionItem:'',
    inspectionMethod:0,
    standardRange:'',
    isPhoto:0,
    id:'',
    picUrl1:'',
    picUrl2:'',
    remark:''
  }]
  @State inspectPlan:InspectPlan = {
    planName:'',
    normGroupId:'',
    stationId:''
  }
  @State inspectTask:Task = {
    actualStartTime:'',
    actualEndTime:'',
    taskStatus:0,
    id:'',
  }
  async aboutToAppear(){
    class RouTmp {
      inspectionPlanId: string = ''
      taskId:string = ''
    }
    const params: RouTmp = router.getParams() as RouTmp; // 获取传递过来的参数
对象
    this.inspectionPlanId = params.inspectionPlanId
    this.taskId=params.taskId
    this.inspectPlan = await this.getInspectPlanDetail()
    this.inspectTask = await this.getInspectTask()
    const selectedOption = this.taskStatusOptions.find(
      item => item.value === this.inspectTask.taskStatus)
    this.status = this.inspectTask.taskStatus
    this.taskStatus = selectedOption?.label ?? '默认值'
    this.btnTxt =  this.inspectTask.taskStatus==0?'开始巡检
':(this.inspectTask.taskStatus==1?'完成巡检':'')
    const inspectStandardList=await
this.getInspectStandard(this.inspectPlan.normGroupId,this.taskId,this.inspe
ctionPlanId)
    this.inspectStandardList =
inspectStandardList.map((item):InspectStandard=>{
      const inspectionMethod =
this.inspectionMethodOptions.find(i=>i.value==item.inspectionMethod)?.label
      return {
        inspectionItem:item.inspectionItem,
        inspectionMethod:inspectionMethod?inspectionMethod:'默认值',
        standardRange:item.standardRange,
        isPhoto:item.isPhoto,
        id:item.id,
        picUrl1:item.picUrl1,
        picUrl2:item.picUrl2,
        remark:item.remark
      }
    })
    for (let index = 0; index < inspectStandardList.length; index++) {
      this.imgObject.push(new KeyValue(inspectStandardList[index].id,[]))
    }
```

```
    for (let index = 0; index < inspectStandardList.length; index++) {
      if(inspectStandardList[index].picUrl1){

this.imgObject.find(item=>item.key==inspectStandardList[index].id)?.value.p
ush(inspectStandardList[index].picUrl1)
      }
      if(inspectStandardList[index].picUrl2){

this.imgObject.find(item=>item.key==inspectStandardList[index].id)?.value.p
ush(inspectStandardList[index].picUrl2)
      }
    }
    this.stationList = await this.getStationList()
    this.stationName =
this.stationList.find(item=>item.id==this.inspectPlan.stationId)?.name
  }
  deleteImage(id:string,index:number){
    this.imgObject.find(item=>item.key==id)?.value.splice(index,1)
    this.imgObject = [...this.imgObject]
  }
  selectImage(id:string,index:number) {
    fileSelect().then((uri: string) => {
      if(uri){
        this.imageUri = uri || '';
        this.picList.push(this.imageUri)
        this.imgObject.find(item=>item.key==id)?.value.push(this.imageUri)
        this.imgObject = [...this.imgObject]
      }
    });
  }
  async upload(url:string){
    return await fileUpload(getContext(this), url)
  }
  async submit(){
    if(this.inspectTask.taskStatus==0){
      this.inspectTask.taskStatus=1
    }else if(this.inspectTask.taskStatus==1){
      this.inspectTask.taskStatus=3
    }
    let opsInspectionNormPicVO:Array<Norm> = []
    for (let index = 0; index < this.imgObject.length; index++) {
      let picUrl1 = ''
      let picUrl2 = ''
      if(this.imgObject[index].value[0]){
        picUrl1 = await this.upload(this.imgObject[index].value[0])
      }
      if(this.imgObject[index].value[1]){
        picUrl2 = await this.upload(this.imgObject[index].value[1])
      }
      opsInspectionNormPicVO.push({
         normId:this.imgObject[index].key,
         picUrl1:picUrl1,
         picUrl2:picUrl2,
         remark:this.remarkList[index]
       })
    }
    this.inspectTask.opsInspectionNormPicVO = opsInspectionNormPicVO
    httpRequestPost(Constants.SERVER +
      '/ml-mlink/inspectiontask/update',this.inspectTask)
      .then((result: ResponseResult) => {
        if (result && result.code === 200) {
          router.back()
        } else {
        }
```

```
      })
      .catch((err: Error) => {
      });
  }
  getInspectTask(): Promise<Task>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        `/ml-mlink/inspectiontask/detail?id=${this.taskId}`)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getInspectPlanDetail(): Promise<InspectPlan>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        `/ml-mlink/inspectionplan/detail?id=${this.inspectionPlanId}`)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getInspectStandard(id:string,taskId:string,planId:string):
Promise<InspectStandard[]>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
`/ml-mlink/opsInspectionNorm/list?normGroupId=${id}&inspectionTaskId=${task
Id}&inspectionPlanId=${planId}`)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getStationList(): Promise<Station[]>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        '/ml-mlink/station/list')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
```

```
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  build() {
    Scroll(){
      Column(){
        Column(){
          Row(){
            Text('名称').fontColor(0x0081FF).fontSize(20)
            Text(this.inspectPlan.planName).fontSize(20)
          }
          .width('100%')
          .justifyContent(FlexAlign.SpaceBetween)
          .padding(15)
          Divider().strokeWidth(8).color('#F1F3F5')
          Row(){
            Text('当前状态').fontColor(0x0081FF).fontSize(20)
            Text(this.taskStatus).fontSize(20)
          } .width('100%')
          .justifyContent(FlexAlign.SpaceBetween)
          .padding(15)
          Divider().strokeWidth(8).color('#F1F3F5')
          Row(){
            Text('站点').fontColor(0x0081FF).fontSize(20)
            Text(this.stationName).fontSize(20)
          }.width('100%')
          .justifyContent(FlexAlign.SpaceBetween)
          .padding(15)
          Divider().strokeWidth(8).color('#F1F3F5')
          Row(){
            Text('任务开始时间').fontColor(0x0081FF).fontSize(20)
            Text(this.inspectTask.actualStartTime).fontSize(20)
          }.width('100%')
          .onClick(()=>{
            DatePickerDialog.show({
              start: new Date("2000-1-1"),
              end: new Date("2100-12-31"),
              selected: new Date(this.inspectTask.actualStartTime),
              showTime:true,
              useMilitaryTime:true,
              onDateAccept: (value: Date) => {
                this.inspectTask.actualStartTime = formatDate(value)
              },
              onCancel: () => {
                console.info("DatePickerDialog:onCancel()")
              },
              onDateChange: (value: Date) => {
                console.info("DatePickerDialog:onDateChange()" +
value.toString())
              },
            })
          })
          .justifyContent(FlexAlign.SpaceBetween)
          .padding(15)
          Divider().strokeWidth(8).color('#F1F3F5')
          Row(){
            Text('任务结束时间').fontColor(0x0081FF).fontSize(20)
            Text(this.inspectTask.actualEndTime).fontSize(20)
          }.width('100%')
          .justifyContent(FlexAlign.SpaceBetween)
          .padding(15)
```

```
          .onClick(()=>{
            DatePickerDialog.show({
              start: new Date("2000-1-1"),
              end: new Date("2100-12-31"),
              selected: new Date(this.inspectTask.actualEndTime),
              showTime:true,
              useMilitaryTime:true,
              onDateAccept: (value: Date) => {
                this.inspectTask.actualEndTime = formatDate(value)
              },
              onCancel: () => {
                console.info("DatePickerDialog:onCancel()")
              },
              onDateChange: (value: Date) => {
                console.info("DatePickerDialog:onDateChange()" +
value.toString())
              },
            })
          })
          Divider().strokeWidth(8).color('#F1F3F5')
        }
        .backgroundColor(0xFFFFFF)
        .width('100%')
        .margin({
          top:10
        })
        Column(){
          Row(){
            Text('巡检标准')
              .fontColor(0x0081FF).fontSize(20)
          }
          .justifyContent(FlexAlign.Start)
          .padding(15)
          .width('100%')
          ForEach(this.inspectStandardList, (item: InspectStandard,index) => {
            Row(){
              Column(){
                Text((index+1).toString()+'、')
              }
              Column(){
                Row(){
                  Text('项目:')
                  Text(item.inspectionItem)
                }.justifyContent(FlexAlign.Start)
                .width('100%')
                .padding({
                  bottom:5
                })
                Row(){
                  Text('方法:')
                  Text(item.inspectionMethod as string)
                }.justifyContent(FlexAlign.Start)
                .width('100%')
                .padding({
                  bottom:5
                })
                Row(){
                  Text('是否拍照:')
                  Text(item.isPhoto?'是':'否')
                }.justifyContent(FlexAlign.Start)
                .width('100%')
                .padding({
                  bottom:5
                })
```

```
                Row(){
                  Text('标准:')
                  Text(item.standardRange)
                }.justifyContent(FlexAlign.Start)
                .width('100%')
                .padding({
                  bottom:5
                })
                if(this.status !==0){
                  Row(){
                    if(this.status==1){
                      Text('上传巡检图片')
                        .fontColor(0x0081FF).fontSize(20)
                    }
                    if(this.status==3){
                      Text('巡检图片')
                        .fontColor(0x0081FF).fontSize(20)
                    }
                  }
                  .justifyContent(FlexAlign.Start)
                  .width('100%')
                  Row() {
                    Flex({ wrap: FlexWrap.Wrap }) {
                      ForEach(this.imgObject.find(i=>i.key==item.id)?.value,
(i: string,index) => {
                        Stack({ alignContent: Alignment.TopEnd}){
                          Image(i).width(100)
                            .height(100)
                          if(this.status==1) {
                            Text('X')
                              .fontColor(0xFFFFFF)
                              .fontSize(18)
                              .margin({
                                top:3,
                                right:3
                              })
                              .onClick(()=>{
                                this.deleteImage(item.id,index)
                              })
                          }
                        }
                        .padding({
                          top:10
                        })
                      }, (i: string) => i)
                      if(this.status==1){
                        Image($r('app.media.ic_add_pic')).width(100)
                          .height(100).margin({
                          top:10
                        })
                          .onClick(() => this.selectImage(item.id,index))
                      }
                    }
                  } .width('100%')
                  Row(){
                    Text('巡检结论')
                      .fontColor(0x0081FF).fontSize(20)
                  }
                  .justifyContent(FlexAlign.Start)
                  .width('100%')
                  .margin({
                    top:10
                  })
                  Row(){
```

```
                    TextArea({placeholder:'请输入巡检结论
',text:item.remark}).width(300)
                      .onChange((value: string) => {
                        this.remarkList[index] = value
                      })
                  }     .width('100%').margin({
                    top:10
                  })
                }
              }
            }.alignItems(VerticalAlign.Top)
            .justifyContent(FlexAlign.Start)
            .width('100%')
            .padding(15)
            Divider().strokeWidth(8).color('#F1F3F5')
          }, (item: InspectStandard) => item.id)
          if(this.btnTxt){
            Row(){
              Button(this.btnTxt, { type: ButtonType.Normal, stateEffect:
true })
                .borderRadius(8)
                .backgroundColor(0x317aff)
                .width('100%')
                .height(40)
            }.padding({
              top:20,
              bottom:20,
              left:10,
              right:10
            })
            .onClick(()=>{
              this.submit()
            })
          }
        }
        .backgroundColor(0xFFFFFF)
      }.backgroundColor(0xF5F5F5)
    }
  }
}
@Entry
@Component
struct Repair {
  @State userId: string = '';
  pageSize: number = 20
  page1: number = 1
  page2: number = 1
  page3: number = 1
  @State type: string = ''
  @State fontColor: string = '#182431'
  @State selectedFontColor: string = '#007DFF'
  @State currentIndex: number = 0
  @State selectedIndex: number = 0
  @State repairList1: Info[] = []
  @State repairList2: Info[] = []
  @State repairList3: Info[] = []
  hasMore1:boolean = false
  hasMore2:boolean = false
  hasMore3:boolean = false
  private controller: TabsController = new TabsController()
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  async getPreference(key: string) {
```

```
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    return (await preference.get(key, '')).toString();
  }
  async aboutToAppear() {
    class RouTmp {
      type: string = ''
    }
    const params: RouTmp = router.getParams() as RouTmp; // 获取传递过来的参数
对象
    this.type = params.type
  }
  async onPageShow() {
    this.page1=1
    this.page2=1
    this.page3=1
    this.repairList1=[]
    this.repairList2=[]
    this.repairList3=[]
    this.userId = await this.getPreference('userid');
    let queryParameter1: QueryData = {
      size: this.pageSize,
      current: this.page1,
      taskStatus: 0
    }
    let queryParameter2: QueryData = {
      size: this.pageSize,
      current: this.page2,
      taskStatus: 1
    }
    let queryParameter3: QueryData = {
      size: this.pageSize,
      current: this.page3,
      taskStatus: 3
    }
    if (this.type == 'repair') {
      queryParameter1.assignedPersonId = this.userId
      queryParameter2.assignedPersonId = this.userId
      queryParameter3.assignedPersonId = this.userId
    } else {
      queryParameter1.createUser = this.userId
      queryParameter2.createUser = this.userId
      queryParameter3.createUser = this.userId
    }
    let data1 = await this.getRepairList(queryParameter1)
    let data2 = await this.getRepairList(queryParameter2)
    let data3 = await this.getRepairList(queryParameter3)
    this.repairList1 = data1.records
    if(this.repairList1.length==data1.size){
      this.page1++
      this.hasMore1 = true
    }
    this.repairList2 = data2.records
    if(this.repairList2.length==data2.size){
      this.page2++
      this.hasMore2 = true
    }
    this.repairList3 = data3.records
    if(this.repairList3.length==data3.size){
      this.page3++
      this.hasMore3 = true
    }
  }
```

```
  getRepairList(queryData: QueryData): Promise<QueryModel<Info>> {
    return new Promise(async (resolve: Function, reject: Function) => {
      if (this.type == 'repair') {
        httpRequestGet(Constants.SERVER +

`/ml-mlink/repairtask/page?current=${queryData.current}&size=${queryData.si
ze}&taskStatus=${queryData.taskStatus}&assignedPersonId=${queryData.assigne
dPersonId}`)
          .then((result: ResponseResult) => {
            if (result && result.code === 200) {
              resolve(result.data);
            } else {
              reject('wrong');
            }
          })
          .catch((err: Error) => {
            reject('error');
          });
      }else {
        httpRequestGet(Constants.SERVER +

`/ml-mlink/repairtask/page?current=${queryData.current}&size=${queryData.si
ze}&taskStatus=${queryData.taskStatus}&createUser=${queryData.createUser}`)
          .then((result: ResponseResult) => {
            if (result && result.code === 200) {
              resolve(result.data);
            } else {
              reject('wrong');
            }
          })
          .catch((err: Error) => {
            reject('error');
          });
      }
    });
  }
  updateRepairList(queryData: QueryData){
    if(this.currentIndex==0){
      if(this.hasMore1){
        this.getRepairList(queryData).then((result)=>{
          let data = result.records
          if(data.length==result.size){
            this.page1++
            this.hasMore1 = true
          }else {
            this.hasMore1 = false
          }
          this.repairList1 = this.repairList1.concat(data)
        })
      }else {
        AlertDialog.show(
          {
            title: '提示',
            message: '已没有更多数据',
            autoCancel: true,
            alignment: DialogAlignment.Center,
            confirm: {
              value: '确定',
              action: () => {
                console.info('Button-clicking callback')
              }
            },
            cancel: () => {
              console.info('Closed callbacks')
```

```
          },
        })
    }
  }else  if(this.currentIndex ==1 ){
    if(this.hasMore2){
      this.getRepairList(queryData).then((result)=>{
        let data = result.records
        if(data.length==result.size){
          this.page2++
          this.hasMore2 = true
        }else {
          this.hasMore2 = false
        }
        this.repairList2 = this.repairList2.concat(data)
      })
    }else {
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
              console.info('Button-clicking callback')
            }
          },
          cancel: () => {
            console.info('Closed callbacks')
          },
        })
    }
  }else {
    if(this.hasMore3){
      this.getRepairList(queryData).then((result)=>{
        let data = result.records
        if(data.length==result.size){
          this.page3++
          this.hasMore3 = true
        }else {
          this.hasMore3 = false
        }
        this.repairList3= this.repairList3.concat(data)
      })
    }else {
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
              console.info('Button-clicking callback')
            }
          },
          cancel: () => {
            console.info('Closed callbacks')
          },
        })
    }
  }
```

```
  }
  @Builder
  tabBuilder(index: number, name: string) {
    Column() {
      Text(name)
        .fontColor(this.selectedIndex === index ? this.selectedFontColor :
this.fontColor)
        .fontSize(16)
        .fontWeight(this.selectedIndex === index ? 500 : 400)
        .lineHeight(22)
        .margin({ top: 17, bottom: 7 })
      Divider()
        .strokeWidth(2)
        .color('#007DFF')
        .opacity(this.selectedIndex === index ? 1 : 0)
    }.width('100%')
  }
  @Builder
  repairList(params: param) {
    Scroll() {
      Column() {
        ForEach(params.list, (item: Info) => {
          Row() {
            if (item.taskStatus == 0) {
              Column() {
                Text('未检').fontColor(0xFFFFFF)
              }.backgroundColor(0xFF6766).width(50).height('100%').justifyCo
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            } else if (item.taskStatus == 1) {
              Column() {
                Text('进行中').fontColor(0xFFFFFF)
              }.backgroundColor(0x47E37E).width(50).height('100%').justifyCo
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            } else {
              Column() {
                Text('已完成').fontColor(0xFFFFFF)
              }.backgroundColor(0x4B93E7).width(50).height('100%').justifyCo
ntent(FlexAlign.Center)
              .padding({
                left:10,
                right:10
              })
              .borderRadius({
                topLeft:20,
                bottomLeft:20
              })
            }
            Column() {
```

```
            Text(item.createTime)
            Text(item.name).margin({
              top:15
            })
          }.layoutWeight(1).alignItems(HorizontalAlign.Start).margin({
            left:10,
          })
          Image($r('app.media.arrow')).width(20)
        }.height(100)
        .justifyContent(FlexAlign.Center)
        .margin({
          top:10,
          left:10,
          right:10
        })
        .padding({
          right:15
        })
        .onClick(() => {
          router.pushUrl({
            url: 'pages/RepairDetail',
            params:{
              id:item.id
            }
          });
        })
        .alignItems(VerticalAlign.Center)
        .backgroundColor(0xFFFFFF)
        .borderRadius(20)
        Divider().strokeWidth(3).color('#F1F3F5')
      }, (item: Info) => item.id)
    }.width('100%')
  }.layoutWeight(1).align(Alignment.Top).height('100%').onScrollEdge((sid
e: Edge) => {
    if (side == Edge.Bottom) {
      if(this.currentIndex==0){
        let queryParameter1: QueryData = {
          size: this.pageSize,
          current: this.page1,
          taskStatus: 0
        }
        if (this.type == 'repair') {
          queryParameter1.assignedPersonId = this.userId
        } else {
          queryParameter1.createUser = this.userId
        }
        this.updateRepairList(queryParameter1)
      }else if(this.currentIndex==1){
        let queryParameter2: QueryData = {
          size: this.pageSize,
          current: this.page2,
          taskStatus: 1
        }
        if (this.type == 'repair') {
          queryParameter2.assignedPersonId = this.userId
        } else {
          queryParameter2.createUser = this.userId
        }
        this.updateRepairList(queryParameter2)
      }
      else {
        let queryParameter3: QueryData = {
          size: this.pageSize,
          current: this.page3,
```

```
          taskStatus: 3
        }
      if (this.type == 'repair') {
        queryParameter3.assignedPersonId = this.userId
      } else {
        queryParameter3.createUser = this.userId
      }
        this.updateRepairList(queryParameter3)
      }
    }
  })
 }
 build() {
   Column() {
     Tabs({ barPosition: BarPosition.Start, index: this.currentIndex,
controller: this.controller }) {
       TabContent() {
         this.repairList({ list: this.repairList1 })
       }.tabBar(this.tabBuilder(0, '未检'))
       TabContent() {
         this.repairList({ list: this.repairList2 })
       }.tabBar(this.tabBuilder(1, '进行中'))
       TabContent() {
         this.repairList({ list: this.repairList3 })
       }.tabBar(this.tabBuilder(2, '已完成'))
     }
     .vertical(false)
     .barMode(BarMode.Fixed)
     .barWidth(360)
     .barHeight(56)
     .animationDuration(400)
     .onChange((index: number) => {
       // currentIndex 控制 TabContent 显示页签
       this.currentIndex = index
     })
     .onAnimationStart((index: number, targetIndex: number, event:
TabsAnimationEvent) => {
       if (index === targetIndex) {
         return
       }
       // selectedIndex 控制自定义 TabBar 内 Image 和 Text 颜色切换
       this.selectedIndex = targetIndex
     })
     .width('100%')
     .height('100%')
     .backgroundColor('#F1F3F5')
   }.width('100%')
 }
}
import { router } from '@kit.ArkUI';
import Constants from '../common/constant/Constants';
import { ResponseResult } from '../viewmodel/ResponseResult';
import { httpRequestGet, httpRequestPost,RepairTask } from
'../common/utils/HttpUtil'
import {LengthMetrics} from '@kit.ArkUI';
import { fileSelect,fileUpload } from '../common/utils/FileUtil';
interface Station {
 name:string,
 id:string
}
interface Device {
 deviceName:string,
 id:string
}
```

```
interface Option {
  label:string,
  value:number
}
function formatDate(date: Date): string {
  const year = date.getFullYear();
  const month = String(date.getMonth() + 1).padStart(2, '0'); // 月份从 0 开
始，需要加 1
  const day = String(date.getDate()).padStart(2, '0');
  const hours = String(date.getHours()).padStart(2, '0');
  const minutes = String(date.getMinutes()).padStart(2, '0');
  const seconds = String(date.getSeconds()).padStart(2, '0');
  return `${year}-${month}-${day} ${hours}:${minutes}:${seconds}`;
}
@Entry
@Component
struct RepairDetail {
  @State message: string = 'RepairDetail';
  @State repairId:string = ''
  @State taskStatus:string=''
  @State status:number = 0
  @State btnTxt:string=''
  @State repairTask:RepairTask = {
    picUrl:'',
    repairPicUrl:'',
    problemDes:'',
    taskStatus:0,
    deviceId:'',
    stationId:'',
    id:'',
    repairStart:'',
    repairEnd:''
  }
  @State stationList:Station[]=[{name:'',id:''}]
  @State deviceList:Device[]=[{deviceName:'',id:''}]
  @State deviceName:string | undefined=''
  @State stationName:string | undefined = ''
  @State picArr:string[]=[]
  @State repairPicArr:string[]=[]
  @State picList:string[]=[];
  @State imageUri: string = '';
  taskStatusOptions:Option[]= [{
    label: '未检',
    value: 0
  },
    {
      label: '进行中',
      value: 1
    },
    {
      label: '暂停',
      value: 2
    },
    {
      label: '已检',
      value: 3
    },
    {
      label: '已取消',
      value: 4
    }
  ]
  async aboutToAppear(){
    class RouTmp {
```

```
      id: string = ''
    }
    const params: RouTmp = router.getParams() as RouTmp; // 获取传递过来的参数
对象
    this.repairId = params.id
    this.repairTask = await this.getRepairDetail()
    this.status = this.repairTask.taskStatus
    if(this.repairTask.picUrl){
      this.picArr = this.repairTask.picUrl.split(',')
    }
    if(this.repairTask.repairPicUrl){
      this.repairPicArr = this.repairTask.repairPicUrl.split(',')
    }
    const selectedOption = this.taskStatusOptions.find(
      item => item.value === this.repairTask.taskStatus)
    this.taskStatus = selectedOption?.label ?? '默认值'
    this.btnTxt =  this.repairTask.taskStatus==0?'开始维修
':(this.repairTask.taskStatus==1?'完成维修':'')
    this.stationList = await this.getStationList()
    this.deviceList = await this.getDeviceList()
    this.deviceName =
this.deviceList.find(item=>item.id==this.repairTask.deviceId)?.deviceName
    this.stationName =
this.stationList.find(item=>item.id==this.repairTask.stationId)?.name
  }
  async upload(url:string){
    return await fileUpload(getContext(this), url)
  }
  deleteImage(index:number){
    this.picList.splice(index,1)
  }
  async submit(){
    if(this.repairTask.taskStatus==0){
      this.repairTask.taskStatus=1
    }else if(this.repairTask.taskStatus==1){
      this.repairTask.taskStatus=3
    }
    let repairPicUrl:string[] | string = []
    for (let index = 0; index < this.picList.length; index++) {
      let url = await this.upload(this.picList[index])
      repairPicUrl.push(url)
    }
    repairPicUrl=repairPicUrl.join(',')
    this.repairTask.repairPicUrl = repairPicUrl
    httpRequestPost(Constants.SERVER +
      '/ml-mlink/repairtask/update',this.repairTask)
      .then((result: ResponseResult) => {
              if (result && result.code === 200) {
        router.back()
       } else {
       }
      })
      .catch((err: Error) => {
      });
  }
  getStationList(): Promise<Station[]>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        '/ml-mlink/station/list')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
```

```
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
getDeviceList(): Promise<Device[]>{
  return new Promise(async (resolve: Function, reject: Function) => {
    httpRequestGet(Constants.SERVER +
      '/ml-mlink/opsDeviceManager/list')
      .then((result: ResponseResult) => {
        if (result && result.code === 200) {
          resolve(result.data);
        } else {
          reject('wrong');
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
getRepairDetail(): Promise<RepairTask>{
  return new Promise(async (resolve: Function, reject: Function) => {
    httpRequestGet(Constants.SERVER +
      `/ml-mlink/repairtask/detail?id=${this.repairId}`)
      .then((result: ResponseResult) => {
        if (result && result.code === 200) {
          resolve(result.data);
        } else {
          reject('wrong');
        }
      })
      .catch((err: Error) => {
        reject('error');
      });
  });
}
selectImage() {
  fileSelect().then((uri: string) => {
    if(uri){
      this.imageUri = uri || '';
      this.picList.push(this.imageUri)
    }
  });
}
build() {
  Scroll(){
    Column(){
      Row(){
        Text('设备').fontColor(0x0081FF).fontSize(20)
        Text(this.deviceName)
          .fontSize(20)
      }
      .width('100%')
      .justifyContent(FlexAlign.SpaceBetween)
      .padding(15)
      Divider().strokeWidth(8).color('#F1F3F5')
      Row(){
        Text('当前状态').fontColor(0x0081FF).fontSize(20)
        Text(this.taskStatus).fontSize(20)
      } .width('100%')
      .justifyContent(FlexAlign.SpaceBetween)
```

```
        .padding(15)
        Divider().strokeWidth(8).color('#F1F3F5')
        Row(){
          Text('站点').fontColor(0x0081FF).fontSize(20)
          Text(this.stationName).fontSize(20)
        }.width('100%')
        .justifyContent(FlexAlign.SpaceBetween)
        .padding(15)
        Divider().strokeWidth(8).color('#F1F3F5')
        Row(){
          Text('任务开始时间').fontColor(0x0081FF).fontSize(20)
          Text(this.repairTask.repairStart).fontSize(20)
        }.width('100%')
        .onClick(()=>{
          DatePickerDialog.show({
            start: new Date("2000-1-1"),
            end: new Date("2100-12-31"),
            selected: new Date(this.repairTask.repairStart),
            showTime:true,
            useMilitaryTime:true,
            onDateAccept: (value: Date) => {
              this.repairTask.repairStart = formatDate(value)
            },
            onCancel: () => {
              console.info("DatePickerDialog:onCancel()")
            },
            onDateChange: (value: Date) => {
              console.info("DatePickerDialog:onDateChange()" +
value.toString())
            },
          })
        })
        .justifyContent(FlexAlign.SpaceBetween)
        .padding(15)
        Divider().strokeWidth(8).color('#F1F3F5')
        Row(){
          Text('任务结束时间').fontColor(0x0081FF).fontSize(20)
          Text(this.repairTask.repairEnd).fontSize(20)
        }.width('100%')
        .onClick(()=>{
          DatePickerDialog.show({
            start: new Date("2000-1-1"),
            end: new Date("2100-12-31"),
            selected: new Date(this.repairTask.repairEnd),
            showTime:true,
            useMilitaryTime:true,
            onDateAccept: (value: Date) => {
              this.repairTask.repairEnd = formatDate(value)
            },
            onCancel: () => {
              console.info("DatePickerDialog:onCancel()")
            },
            onDateChange: (value: Date) => {
              console.info("DatePickerDialog:onDateChange()" +
value.toString())
            },
          })
        })
        .justifyContent(FlexAlign.SpaceBetween)
        .padding(15)
        Divider().strokeWidth(8).color('#F1F3F5')
        Row(){
          Text('问题描述').fontColor(0x0081FF).fontSize(20)
        }.width('100%') .padding(15).justifyContent(FlexAlign.Start)
```

```
        Row(){
          Text(this.repairTask.problemDes)
        }.width('100%')
        .padding({
          left:15,
          right:15,
          bottom:15
        }).justifyContent(FlexAlign.Start)
        Divider().strokeWidth(8).color('#F1F3F5')
        Row(){
          Text('故障图片').fontColor(0x0081FF).fontSize(20)
        }.width('100%') .padding(15).justifyContent(FlexAlign.Start)
        Row() {
          Flex({  wrap: FlexWrap.Wrap }) {
            ForEach(this.picArr, (item: string,index) => {
              Image(item).width(100)
                .height(100).margin({
                left:10,
              })
            })
          }.padding({
            left:5,
            bottom:15,
            right:15
          })
        }.width('100%').justifyContent(FlexAlign.Center)
        if (this.status==3){
          Row(){
            Text('维修图片').fontColor(0x0081FF).fontSize(20)
          }.width('100%') .padding(15).justifyContent(FlexAlign.Start)
          Row() {
            Flex({  wrap: FlexWrap.Wrap }) {
              ForEach(this.repairPicArr, (item: string,index) => {
                Image(item).width(100)
                  .height(100).margin({
                  left:10,
                })
              })
            }.padding({
              left:5,
              bottom:15,
              right:15
            })
          }.width('100%').justifyContent(FlexAlign.Center)
        }
        if (this.status==1){
          Divider().strokeWidth(8).color('#F1F3F5')
          Row(){
            Text('上传维修图片').fontColor(0x0081FF).fontSize(20)
          }.width('100%') .padding(15).justifyContent(FlexAlign.Start)
          Row() {
            Flex({ wrap: FlexWrap.Wrap }) {
              ForEach(this.picList, (item: string,index) => {
                Stack({ alignContent: Alignment.TopEnd}){
                  Image(item).width(100)
                    .height(100).margin({
                    left:10,
                  })
                  Text('X')
                    .fontColor(0xFFFFFF)
                    .fontSize(18)
                    .margin({
                      top:3,
                      right:3
```

```
                    })
                    .onClick(()=>{
                      this.deleteImage(index)
                    })
                  }
                }, (item: string) => item)
                Image($r('app.media.ic_add_pic')).width(100)
                  .height(100).margin({
                  left:10,
                })
                  .onClick(() => this.selectImage())
              }
            }
            .padding({
              left:5,
              bottom:15,
              right:15
            })
          }
          if(this.btnTxt){
            Row(){
              Button(this.btnTxt, { type: ButtonType.Normal, stateEffect: true })
                .borderRadius(8)
                .backgroundColor(0x317aff)
                .width('100%')
                .height(40)
            }.padding({
              top:20,
              bottom:20,
              left:10,
              right:10
            })
            .onClick(()=>{
              this.submit()
            })
          }
        }
        .backgroundColor(0xFFFFFF)
        .width('100%')
        .margin({
          top:10
        })
      }
    }
}
@Entry
@Component
export struct RepairInfo {
  @State personIndex: number = 0
  @State deviceIndex: number = 0
  @State stationIndex: number = 0
  @State personList: Array<SelectItem> = []
  @State deviceList: Array<SelectItem> = []
  @State stationList: Array<SelectItem> = []
  @State person: string = ''
  @State device: string = ''
  @State station: string = ''
  @State description: string = ''
  @State imageUri: string = '';
  @State picUrl:string='';
  @State picList:string[]=[];
  @State beginDate:string=''
  @State endDate:string = ''
  @State isUploading: boolean = false;
```

```
  uploadPicList:string[]=[]
  time: number = 10
  personId: string = ''
  deviceId: string = ''
  stationId: string = ''
  selectedDate: Date =  new Date();
  @Styles
  select() {
    .width('100%')
    .padding({
      left: 20,
      right: 20,
      bottom: 10,
      top: 10
    })
    .border({
      width: {
        bottom: 1
      },
      color: { bottom: Color.Green },
      style: {
        bottom: BorderStyle.Solid
      }
    })
  }
  async aboutToAppear() {
    this.beginDate = formatDate(new Date())
    this.endDate = formatDate(new Date())
    this.getEquipment().then(res => {
      console.log(JSON.stringify(res));
      this.deviceList = res.map((item: Device): SelectItem => {
        return {
          value: item.deviceName,
          id: item.id
        };
      });
      this.device = this.deviceList[0].value
      this.deviceId = this.deviceList[0].id
      console.log(JSON.stringify(this.deviceList))
    })
    this.getStation().then(res => {
      console.log(JSON.stringify(res));
      this.stationList = res.map((item: Station): SelectItem => {
        return {
          value: item.name,
          id: item.id
        };
      });
      this.station = this.stationList[0].value
      this.stationId = this.stationList[0].id
      console.log(JSON.stringify(this.stationList))
    })
    this.getPerson().then(res => {
      console.log(JSON.stringify(res));
      this.personList = res.map((item: Person): SelectItem => {
        return {
          value: item.name,
          id: item.id
        };
      });
      this.person = this.personList[0].value
      this.personId = this.personList[0].id
      console.log(JSON.stringify(this.personList))
    })
```

```
  }
  getPerson(): Promise<Person[]> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER+'/blade-user/list')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getStation(): Promise<Station[]> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER+'/ml-mlink/station/list')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getEquipment(): Promise<Device[]> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER+'/ml-mlink/opsDeviceManager/list')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  selectImage() {
    fileSelect().then((uri: string) => {
      if(uri){
        this.imageUri = uri || '';
        this.picList.push(this.imageUri)
      }
    });
  }
  deleteImage(index:number){
    this.picList.splice(index,1)
  }
  async upload(){
    for (let index = 0; index < this.picList.length; index++) {
      this.picUrl = await fileUpload(getContext(this), this.picList[index])
      this.uploadPicList.push(this.picUrl)
    }
    let repairInfo:RepairData = {
      assignedPersonId:this.personId,
```

```
      deviceId:this.deviceId,
      stationId:this.stationId,
      cycleStart:this.beginDate,
      cycleEnd:this.endDate,
      picUrl:this.uploadPicList.join(','),
      problemDes:this.description,
      sendTime:this.time
    }
    let code=await this.submit(repairInfo)
    this.isUploading = false;
    AlertDialog.show(
      {
        title: '提示',
        message: '提交成功',
        alignment: DialogAlignment.Center,
        confirm: {
          value: '确认',
          action: () => {
            router.back()
          }
        }
      }
    )
  }
  submit(repairInfo:RepairData):Promise<number>{
    this.isUploading = true;
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestPost(Constants.SERVER+'/ml-mlink/repairtask/save',
repairInfo)
        .then((result: ResponseResult) => {
          resolve(result.code);
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  build() {
    Stack() {
      Column() {
        Row() {
          Text('设备').fontColor(Color.Blue)
          Select(this.deviceList)
            .selected(this.deviceIndex)
            .value(this.device)
            .onSelect((index: number, text?: string) => {
              this.deviceId = this.deviceList[index].id
            })
        }.select()
        .justifyContent(FlexAlign.SpaceBetween)
        Row() {
          Text('责任人').fontColor(Color.Blue)
          Select(this.personList)
            .selected(this.personIndex)
            .value(this.person)
            .onSelect((index: number, text?: string) => {
              this.personId = this.personList[index].id
            })
        }.select()
        .justifyContent(FlexAlign.SpaceBetween)
        Row() {
          Text('站点').fontColor(Color.Blue)
          Select(this.stationList)
            .selected(this.stationIndex)
```

```
            .value(this.station)
            .onSelect((index: number, text?: string) => {
              this.stationId = this.stationList[index].id
            })
        }
        .select()
        .justifyContent(FlexAlign.SpaceBetween)
        Row() {
          Text('工单提前派发时间').fontColor(Color.Blue)
          Row() {
            TextInput({
              text: (this.time).toString()
            }).width('30%').textAlign(TextAlign.Center).type(InputType.Numbe
r)
              .onChange((value: string) => {
                this.time = Number(value)
              })
            Text('分钟').margin({
              left: 10
            })
          }
        }.select()
        .justifyContent(FlexAlign.SpaceBetween)
        Row(){
          Text('计划开始时间').fontColor(Color.Blue)
          Text(this.beginDate).onClick(()=>{
            DatePickerDialog.show({
              start: new Date("2000-1-1"),
              end: new Date("2100-12-31"),
              selected: this.selectedDate,
              showTime:true,
              useMilitaryTime:true,
              onDateAccept: (value: Date) => {
                this.beginDate = formatDate(value)
              },
              onCancel: () => {
                console.info("DatePickerDialog:onCancel()")
              },
              onDateChange: (value: Date) => {
                console.info("DatePickerDialog:onDateChange()" +
value.toString())
              },
            })
          })
        }.select().justifyContent(FlexAlign.SpaceBetween)
        Row(){
          Text('计划结束时间').fontColor(Color.Blue)
          Text(this.endDate).onClick(()=>{
            DatePickerDialog.show({
              start: new Date("2000-1-1"),
              end: new Date("2100-12-31"),
              selected: this.selectedDate,
              showTime:true,
              useMilitaryTime:true,
              onDateAccept: (value: Date) => {
                this.beginDate = formatDate(value)
              },
              onCancel: () => {
                console.info("DatePickerDialog:onCancel()")
              },
              onDateChange: (value: Date) => {
                console.info("DatePickerDialog:onDateChange()" +
value.toString())
              },
```

```
      })
    })
  }.select().justifyContent(FlexAlign.SpaceBetween)
  Row() {
    Text('问题描述').fontColor(Color.Blue)
    TextArea({
      text: this.description,
      placeholder: '请输入问题描述',
    })
      .placeholderFont({ size: 16, weight: 400 })
      .width(336)
      .height(56)
      .margin(20)
      .fontSize(16)
      .backgroundColor('#FFFFFF')
      .onChange((value: string) => {
        this.description = value
      })
  }.select()
  .justifyContent(FlexAlign.SpaceBetween)
  Row() {
    Flex({ wrap: FlexWrap.Wrap }) {
      ForEach(this.picList, (item: string,index) => {
        Stack({ alignContent: Alignment.TopEnd}){
          Image(item).width(100)
            .height(100).margin({
            left:10,
          })
          Text('X')
            .fontColor(0xFFFFFF)
            .fontSize(18)
            .margin({
              top:3,
              right:3
            })
            .onClick(()=>{
              this.deleteImage(index)
            })
        }.padding({
          top:10
        })
      }, (item: string) => item)
      Image($r('app.media.ic_add_pic')).width(100)
        .height(100).margin({
        left:10,
        top:10
      })
        .onClick(() => this.selectImage())
    }
  } .width('100%')
  .padding({
    left: 10,
    right: 20,
    bottom: 10,
    top: 5
  })
  Row(){
    Button('提交')
      .onClick(()=>{
        this.upload()
      })
  }.margin({
    top:30
  })
```

```
      }.width('100%').height('100%')
      .margin({
        top: 30
      })
      if (this.isUploading) {
        UploadingLayout()
      }
    }
  }
}
@Component
export default struct Safe {
  @State token: string = ''
  scroller: Scroller = new Scroller()
  @Prop warnModel: WarnModel;
  @State type: string = '实时'
  @State alarmTime:string = '开始时间'
  @State alarmEndTime:string = '结束时间'
  private select: number | number[] = 0
  private typeList: string[] = ['实时', '历史']
  private beginSelectedDate: Date = new Date()
  private endSelectedDate: Date = new Date()
  @State warnStatistic: WarnStatistic[] = [
    {
      name: '总数',
      count: 0,
      fontColor: 0x62d92a,
      backgroundColor: 0xe5f5d0,
      image: 'app.media.all_num'
    },
    {
      name: '严重',
      count: 0,
      fontColor: 0xfe0000,
      backgroundColor: 0xfed6d6,
      image: 'app.media.warning_hard'
    },
    {
      name: '普通',
      count: 0,
      fontColor: 0xfea52a,
      backgroundColor: 0xfbe8ca,
      image: 'app.media.warning_normal'
    },
    {
      name: '预警',
      count: 0,
      fontColor: 0xfec72e,
      backgroundColor: 0xf9f4d6,
      image: 'app.media.pre_warning'
    }
  ]
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  async getPreference() {
    let token = '';
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    token = (await preference.get('token', '')).toString();
    return token;
  }
  formatDate(date: Date): string {
```

```
    const year = date.getFullYear();
    const month = String(date.getMonth() + 1).padStart(2, '0'); // 月份从 0 开
始，所以要加 1
    const day = String(date.getDate()).padStart(2, '0');
    return `${year}-${month}-${day}`;
  }
  changeWarning?: () => void
  getWarning?:(queryData: SafeQueryData)=> Promise<QueryModel<WarnData>>
  async aboutToAppear() {
    if(this.changeWarning){
      this.changeWarning()
    }
  }
  updateWarning(queryData:SafeQueryData){
    if(this.warnModel.hasMore){
      if(this.getWarning){
        this.getWarning(queryData).then((result) => {
          let data = result.records
          if (data.length === this.warnModel.size) {
            this.warnModel.currentPage++;
            this.warnModel.hasMore = true;
          } else {
            this.warnModel.hasMore = false;
          }
          this.warnModel.warnData = this.warnModel.warnData.concat(data);
          this.warnModel.pageState = PageState.Success
        }).catch((err: string | Resource) => {
        })
      }
    }else {
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
              console.info('Button-clicking callback')
            }
          },
          cancel: () => {
            console.info('Closed callbacks')
          },
        })
    }
  }
  @Builder
  LoadingLayout() {
    CustomRefreshLoadLayout({
      customRefreshLoadClass: new CustomRefreshLoadLayoutClass(true,
        $r('app.media.ic_pull_up_load'), $r('app.string.pull_up_load_text'),
this.warnModel.pullDownRefreshHeight)
    })
  }
  build() {
    Column() {
      Row() {
        Image($r('app.media.title')).width(40)
        Text('当前告警统计')
        Image($r('app.media.title')).width(40).rotate({
          angle: 180
        })
```

```
      }
      .width('100%')
      .backgroundColor(0xFFFFFF)
      .justifyContent(FlexAlign.Center)
      .height(50)
      Row() {
        ForEach(this.warnStatistic, (item: WarnStatistic) => {
          Column() {
            Image($r(item.image)).width(50)

Text(`${item.name}:${item.count}`).fontColor(item.fontColor).margin({
              top: 5
            })
          }
          .width(90)
          .height(90)
          .backgroundColor(item.backgroundColor)
          .borderRadius(20)
          .justifyContent(FlexAlign.Center)
        }, (item: WarnStatistic) => item.name)
      }.backgroundColor(0xffffff)
      .width('100%')
      .justifyContent(FlexAlign.SpaceEvenly)
      Row() {
        TextInput({
          placeholder: '输入查询',
        })
          .placeholderFont({
            size:13
          })
          .fontSize(13)
          .width('25%')
          .textAlign(TextAlign.Center)
          .onChange((value: string) => {
          this.warnModel.alarmContent = value
          this.warnModel.currentPage = 1
          this.warnModel.warnData = []
            this.warnModel.hasMore = true
          this.warnModel.pageState = PageState.Loading
            let queryData:SafeQueryData = {
              current: this.warnModel.currentPage,
              size: this.warnModel.size,
              isRealTime: this.warnModel.isRealTime,
              alarmContent: this.warnModel.alarmContent,
            }
            if(this.warnModel.alarmTime!='开始时间'){
              queryData.alarmTime = this.warnModel.alarmTime+' 00:00:00'
            }
            if(this.warnModel.alarmEndTime!='结束时间'){
              queryData.alarmEndTime = this.warnModel.alarmEndTime+' 23:59:59'
            }
            this.updateWarning(queryData)
        })
        TextInput({
          text: this.warnModel.isRealTime?'实时':'历史'
        })
          .fontSize(13)
          .width('17%')
          .textAlign(TextAlign.Center)
          .focusable(false)
          .onClick(() => {
            TextPickerDialog.show({
              range: this.typeList,
              selected: this.warnModel.isRealTime?0:1,
```

```
            disappearTextStyle: { color: Color.Black, font: { size: 20, weight:
FontWeight.Normal } },
            selectedTextStyle: { color: Color.Blue, font: { size: 30, weight:
FontWeight.Bolder } },
            canLoop: false,
            onAccept: (value: TextPickerResult) => {
              this.select = value.index
              console.log(this.select + '')
              // 点击确定后，被选到的文本数据展示到页面
              this.type = value.value as string
              if (this.select == 0) {
                this.warnModel.isRealTime = 1
              } else {
                this.warnModel.isRealTime = ""
              }
              this.warnModel.currentPage = 1
              this.warnModel.warnData = []
              this.warnModel.pageState = PageState.Loading
              this.warnModel.hasMore = true
              let queryData:SafeQueryData = {
                current: this.warnModel.currentPage,
                size: this.warnModel.size,
                isRealTime: this.warnModel.isRealTime,
                alarmContent: this.warnModel.alarmContent,
              }
              if(this.warnModel.alarmTime!='开始时间'){
                queryData.alarmTime = this.warnModel.alarmTime+' 00:00:00'
              }
              if(this.warnModel.alarmEndTime!='结束时间'){
                queryData.alarmEndTime = this.warnModel.alarmEndTime+'
23:59:59'
              }
                this.updateWarning(queryData)
            },
            onCancel: () => {
              console.info("TextPickerDialog:onCancel()")
            }
          })
        })
      TextInput({
        text: this.warnModel.alarmTime
      })
        .fontSize(13)
        .width('28%')
        .textAlign(TextAlign.Center)
        .focusable(false)
        .onClick(() => {
          CalendarPickerDialog.show({
            selected: this.beginSelectedDate,
            onAccept: (value) => {
              this.beginSelectedDate = value
              this.warnModel.alarmTime = this.formatDate(value)
              this.warnModel.currentPage = 1
              this.warnModel.warnData = []
              this.warnModel.pageState = PageState.Loading
              this.warnModel.hasMore = true
              let queryData:SafeQueryData = {
                current: this.warnModel.currentPage,
                size: this.warnModel.size,
                isRealTime: this.warnModel.isRealTime,
                alarmContent: this.warnModel.alarmContent,
              }
              if(this.warnModel.alarmTime!='开始时间'){
                queryData.alarmTime = this.warnModel.alarmTime+' 00:00:00'
```

```
              }
              if(this.warnModel.alarmEndTime!='结束时间'){
                queryData.alarmEndTime = this.warnModel.alarmEndTime+'
23:59:59'
              }
              this.updateWarning(queryData)
            },
            onCancel: () => {
              console.info("calendar onCancel")
            },
          })
        })
      TextInput({
        text: this.warnModel.alarmEndTime
      })
        .fontSize(13)
        .width('28%')
        .textAlign(TextAlign.Center)
        .focusable(false)
        .onClick(() => {
          CalendarPickerDialog.show({
            selected: this.endSelectedDate,
            onAccept: (value) => {
              this.endSelectedDate=value
              this.warnModel.alarmEndTime = this.formatDate(value)
              this.warnModel.currentPage = 1
              this.warnModel.warnData = []
              this.warnModel.pageState = PageState.Loading
              this.warnModel.hasMore = true
              let queryData:SafeQueryData = {
                current: this.warnModel.currentPage,
                size: this.warnModel.size,
                isRealTime: this.warnModel.isRealTime,
                alarmContent: this.warnModel.alarmContent,
              }
              if(this.warnModel.alarmTime!='开始时间'){
                queryData.alarmTime = this.warnModel.alarmTime+' 00:00:00'
              }
              if(this.warnModel.alarmEndTime!='结束时间'){
                queryData.alarmEndTime = this.warnModel.alarmEndTime+'
23:59:59'
              }
              this.updateWarning(queryData)
            },
            onCancel: () => {
              console.info("calendar onCancel")
            },
          })
        })
    }.width('100%').justifyContent(FlexAlign.SpaceEvenly)
    .height(80)
    .backgroundColor(0xFFFFFF)
    Row() {
      Flex({ direction: FlexDirection.Row }) {
        Text('时间').width('30%').textAlign(TextAlign.Center)
        Text('等级').width('10%').textAlign(TextAlign.Center)
        Text('告警内容').width('50%').textAlign(TextAlign.Center)
      }
    }.width('100%')
    .height(40)
    .alignItems(VerticalAlign.Center)
    .backgroundColor(0xFFFFFF)
    if (this.warnModel.pageState === PageState.Success) {
      Scroll(this.scroller) {
```

```
        Column() {
          ForEach(this.warnModel.warnData, (item: WarnData) => {
            Row() {
              Flex({ direction: FlexDirection.Row }) {

Text(item.alarmTime).width('30%').textAlign(TextAlign.Center)
                if (item.level == 0) {
                  Text('预警
').width('10%').fontColor(0xFEC72E).textAlign(TextAlign.Center)
                } else if (item.level == 1) {
                  Text('普通
').width('10%').fontColor(0xFEA52A).textAlign(TextAlign.Center)
                } else {
                  Text('严重
').width('10%').fontColor(0xFE0000).textAlign(TextAlign.Center)
                }

Text(item.alarmContent).width('50%').textAlign(TextAlign.Center)
                  .maxLines(2)
                  .textOverflow({ overflow: TextOverflow.Ellipsis })
                Row() {
                  Image($r('app.media.arrow')).width('50%')
                }.width('10%')
                .justifyContent(FlexAlign.Center)
              }
            }
            .width('100%')
            .height(60)
            .onClick(()=>{
              router.pushUrl({
                url: 'pages/WarnDetail',
                params:{
                  id:item.id,
                }
              });
            })
            Divider().strokeWidth(3).color('#F1F3F5')
          }, (item: WarnData) => item.id)
        }.padding({
          top: 5,
          bottom: 20
        })
      }.layoutWeight(1).align(Alignment.Top).backgroundColor(0xFFFFFF)
      .onScrollEdge((side: Edge) => {
        if (side == Edge.Bottom) {
          let queryData:SafeQueryData = {
            current: this.warnModel.currentPage,
            size: this.warnModel.size,
            isRealTime: this.warnModel.isRealTime,
            alarmContent: this.warnModel.alarmContent,
          }
          if(this.warnModel.alarmTime!='开始时间'){
            queryData.alarmTime = this.warnModel.alarmTime+' 00:00:00'
          }
          if(this.warnModel.alarmEndTime!='结束时间'){
            queryData.alarmEndTime = this.warnModel.alarmEndTime+' 23:59:59'
          }
          this.updateWarning(queryData)
        }
      })
    } else if (this.warnModel.pageState === PageState.Loading) {
      this.LoadingLayout()
    }
  }
```

```
    .backgroundColor(0xFFFFFF)
    .width('100%')
    .height('100%')
    .justifyContent(FlexAlign.Start)
  }
}
@Entry
@Component
struct WarnDetail {
  @State warnInfoId:string = ''
  @State warnInfo:WarnInfo = {
    address:'',
    alarmContent:'',
    alarmTime:'',
    stationName:'',
    level:0,
    equipmentName:''
  }
  async aboutToAppear() {
    class RouTmp {
      id: string = ''
    }
    const params: RouTmp = router.getParams() as RouTmp; // 获取传递过来的参数
对象
    console.log(JSON.stringify(params))
    this.warnInfoId = params.id
    this.warnInfo = await this.getWarnInfo()
    console.log(JSON.stringify(this.warnInfo))
  }
  eliminateAlarm(){
      httpRequestPost(Constants.SERVER +
        `/ml-mlink/equipmentalarm/oneKeyEliminate`,{id:this.warnInfoId})
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            router.back();
          } else {
          }
        })
        .catch((err: Error) => {
        });
  }
  getWarnInfo(): Promise<WarnInfo>{
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
        `/ml-mlink/equipmentalarm/detail?id=${this.warnInfoId}`)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  build() {
    Column(){
      Column({ space: 5 }){
        Row({ space: 5 }){
          Column() {
            Text('告警站点:').fontColor(0x0081FF)
          }.alignItems(HorizontalAlign.Start)
```

```
        Column(){
          Text(this.warnInfo.stationName)
        }.alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row({ space: 5 }){
        Column() {
          Text('告警地址:').fontColor(0x0081FF)
        }.alignItems(HorizontalAlign.Start)
        Column(){
          Text(this.warnInfo.address)
        }.alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row({ space: 5 }){
        Column() {
          Text('告警时间:').fontColor(0x0081FF)
        }.alignItems(HorizontalAlign.Start)
        Column(){
          Text(this.warnInfo.alarmTime)
        }.alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row({ space: 5 }){
        Column(){
          Text('告警等级:').fontColor(0x0081FF)
        } .alignItems(HorizontalAlign.Start)
        Column(){
          Text(this.warnInfo.level.toString())
        } .alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row({ space: 5 }){
        Column(){
          Text('告警内容:').fontColor(0x0081FF)
        }     .alignItems(HorizontalAlign.Start)
        Column(){
          Text(this.warnInfo.alarmContent)
        }    .alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row({ space: 5 }){
        Column(){
        Text('告警设备:').fontColor(0x0081FF)
        }     .alignItems(HorizontalAlign.Start)
        Column(){
        Text(this.warnInfo.equipmentName)
        }    .alignItems(HorizontalAlign.Start)
        .layoutWeight(1)
      }.alignItems(VerticalAlign.Top)
      Row(){
        Button('消除告警')
          .onClick(()=>{
              this.eliminateAlarm()
          })
      }.alignItems(VerticalAlign.Top)
    }.backgroundColor(0xFFFFFF)
    .alignItems(HorizontalAlign.Start)
    .padding(20)
  } .padding(20).backgroundColor(0xF5F5F5)
  .borderRadius(20)
  .width('100%')
  .height('100%')
}
```

```
}
@Entry
@Component
export default struct Mine {
  @State username:string = ''
  @State avatar:string = ''
  async aboutToAppear() {
    this.username = await this.getPreference('username');
    this.avatar = await this.getPreference('avatar');
  }
  async getPreferencesFromStorage() {
    preference = await preferences.getPreferences(context, 'link');
  }
  async getPreference(key:string) {
    if (!preference) {
      await this.getPreferencesFromStorage();
    }
    return (await preference.get(key, '')).toString();
  }
  build() {
    Scroll(){
      Column(){
        Stack({alignContent:Alignment.TopStart }){

Image('http://120.79.38.76:9000/mlink/upload/20231027/2a41f7a9264de25b91b21
2e36fbdf178.jpg')
            .width('100%')
            .objectFit(ImageFit.Contain)
          Row(){
            Image(this.avatar).width(100).borderRadius(50)
            Text(this.username)
              .margin({
                left:20
              })
          }.margin({
            left:30,
            top:30
          })
        }.width('100%')
        Column() {
          Row(){
            Image($r('app.media.5G')).width(35)
            Text('通信工况')
              .margin({
                left:10
              })
          }.width('90%').height(50)
          .onClick(()=>{
            router.pushUrl({
              url: 'pages/Communication'
            });
          })
          Row(){
            Image($r('app.media.log')).width(35)
            Text('操作日志')
              .margin({
                left:10
              })
          }.width('90%').height(50)
          .onClick(()=>{
            router.pushUrl({
              url: 'pages/Log'
            });
          })
```

```
          Row(){
            Image($r('app.media.setting')).width(35)
            Text('设置')
              .margin({
                left:10
              })
          }.width('90%').height(50)
          .onClick(()=>{
            router.pushUrl({
              url: 'pages/Setting'
            });
          })
          Row(){
            Image($r('app.media.exit')).width(35)
            Text('退出')
              .margin({
                left:10
              })
          }.width('90%').height(50)
          .onClick(()=>{
            router.pushUrl({
              url: "pages/Index"
            })
          })
        }.width('90%').backgroundColor(0xffffff).borderRadius(15)
        .margin({
          top:20,
          bottom:20
        })
      }.width('100%')
      .justifyContent(FlexAlign.Start)
      .height('100%')
    }
    .height('100%')
  }
}
@CustomDialog
struct CustomDialogExample {
  @Link url: string
  controller: CustomDialogController
  cancel: () => void = () => {
  }
  confirm: () => void = () => {
  }
  build() {
    Column() {
      TextInput().onChange((value: string) => {
        this.url = value
      }).height('30%')
      Row() {
        Button('确定').onClick(() => {
          this.controller.close()
          this.confirm()
        })
        Button('取消').onClick(() => {
          this.controller.close()
        })
          .margin({
            left: 10
          })
      }.margin({
        top: 20
      })
    }.height(150).justifyContent(FlexAlign.Center).padding({
```

```
      left: 20,
      right: 20
    })
  }
}
@CustomDialog
struct CustomDialogExample1 {
  @Link urlList: string[]
  @Link selectedUrl: string
  controller: CustomDialogController
  cancel: () => void = () => {
  }
  confirm: () => void = () => {
  }
  build() {
    Column() {
      ForEach(this.urlList, (item: string) => {
        Row() {
          Text(item)
          Radio({ value: item, group: 'radioGroup' })
            .radioStyle({
              checkedBackgroundColor: Color.Blue
            })
            .onChange((isChecked: boolean) => {
              if (isChecked) {
                this.selectedUrl = item
              }
            })
        }
      }, (item: string) => item)
      Row() {
        Button('确定').onClick(() => {
          this.controller.close()
          this.confirm()
        })
        Button('取消').onClick(() => {
          this.controller.close()
        })
          .margin({
            left: 10
          })
      }.margin({
        top: 20
      })
    }.justifyContent(FlexAlign.Center).padding({
      left: 20,
      right: 20
    }).padding({
      top:20,
      bottom:20
    })
  }
}
@Entry
@Component
struct Setting {
  dialogController: CustomDialogController = new CustomDialogController({
    builder: CustomDialogExample({
      cancel: () => {
        this.onCancel()
      },
      confirm: () => {
        this.onAccept()
      },
```

```
      url: $newUrl
    }),
  })
  dialogController1: CustomDialogController = new CustomDialogController({
    builder: CustomDialogExample1({
      cancel: () => {
        this.onCancel1()
      },
      confirm: () => {
        this.onAccept1()
      },
      urlList: $urlList,
      selectedUrl: $selectedUrl
    }),
  })
  @State url: string = ''
  @State urlList: string[] = []
  @State newUrl: string = ''
  @State menu: MenuElement[] = []
  @State selectedUrl: string = ''
  onAccept1() {
    this.urlList = this.urlList.filter(item => item != this.selectedUrl)
    this.menu = this.urlList.map((url: string): MenuElement => ({
      value: url,
      action: () => {
        this.url = url
      }
    }));
    if (dataPreferences) {
      dataPreferences.putSync('urlList', this.urlList);
      dataPreferences.flush((err: BusinessError) => {
        if (err) {
          console.error(`Failed to flush. Code:${err.code},
message:${err.message}`);
          return;
        }
      })
    }
  }
  onCancel1() {
  }
  onAccept() {
    if (dataPreferences) {
      this.urlList.push(this.newUrl)
      dataPreferences.putSync('urlList', this.urlList);
      this.menu = this.urlList.map((url: string): MenuElement => ({
        value: url,
        action: () => {
          this.url = url
        }
      }));
      dataPreferences.flush((err: BusinessError) => {
        if (err) {
          console.error(`Failed to flush. Code:${err.code},
message:${err.message}`);
          return;
        }
      })
    }
  }
  async aboutToAppear() {
    if (dataPreferences) {
      this.url = dataPreferences.getSync('url', Constants.Default_SERVER) as
string;
```

```
      this.urlList = dataPreferences.getSync('urlList', []) as string[]
      this.menu = this.urlList.map((url: string): MenuElement => ({
        value: url,
        action: () => {
          this.url = url
        }
      }));
    }
  }
  build() {
    Column() {
      TextInput({
        text: this.url
      })
        .focusable(false)
        .onChange((value: string) => {
          this.url = value
        })
        .bindMenu(this.menu)
      Row() {
        Button('设置地址').onClick(() => {
          if (dataPreferences) {
            dataPreferences.putSync('url', this.url);
            dataPreferences.flush((err: BusinessError) => {
              if (err) {
                console.error(`Failed to flush. Code:${err.code},
message:${err.message}`);
                return;
              }
              router.clear();
              router.replaceUrl({
                url: 'pages/Index'
              });
            })
          }
        })
        Button('新增地址').margin({
          left: 20
        }).onClick(() => {
          this.dialogController.open()
        })
        Button('删除地址').margin({
          left: 20
        }).onClick(() => {
          this.dialogController1.open()
        })
      }.margin({
        top: 20
      })
    }.padding({
      top: 30,
      left: 20,
      right: 20
    })
  }
}
@Entry
@Component
struct Communication {
  @State message: string = 'Hello World';
  @State wgList: GateWay[] = []
  @State selectList: SelectItem[] = []
  @State currentGw: GateWay = {
    id: '',
```

```
    name: ''
  }
  @State com: number[] = []
  @State comList: Record<number, DeviceInfo[]> = {}
  @State text: string = ''
  @State index: number = 0
  @State gwInfo: GateWayDetail = {
    name: '',
    id: '',
    status: 0,
    comRtus: {}
  }
  async aboutToAppear() {
    this.wgList = await this.getDeviceList()
    this.selectList = this.wgList.map((item): SelectItem => {
      return {
        value: item.name,
        id: item.id
      }
    })
    this.text = this.selectList[0].value
    this.currentGw = this.wgList[0]
    let gateWayInfo: GateWayInfo = {
      gwId: this.currentGw.id,
      gwName: this.currentGw.name ? this.currentGw.name : this.currentGw.id
    }
    this.gwInfo = await this.getDeviceByGwId(gateWayInfo)
    this.comList = this.gwInfo.comRtus
    this.com = Object.keys(this.comList).map(Number)
  }
  getDeviceByGwId(gateWayInfo: GateWayInfo): Promise<GateWayDetail> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
'/ml-mlink/bigscreen/communication/getCommunicationByGwid', gateWayInfo)
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  getDeviceList(): Promise<GateWay[]> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER + '/ml-mlink/datasource/treelist')
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  build() {
      Scroll() {
        Column() {
```

```
        Select(this.selectList)
          .selected(this.index)
          .value(this.text)
          .font({ size: 16, weight: 500 })
          .fontColor('#182431')
          .selectedOptionFont({ size: 16, weight: 400 })
          .optionFont({ size: 16, weight: 400 })
          .menuAlign(MenuAlignType.START, { dx: 0, dy: 0 })
          .optionWidth(200)
          .optionHeight(300)
          .margin({
            top: 20,
            bottom: 20
          })
          .onSelect(async (index: number, text?: string | undefined) => {
            this.index = index;
            if (text) {
              this.text = text;
            } else {
              this.text = ''
            }
            let gateWayInfo: GateWayInfo = {
              gwId: this.selectList[index].id,
              gwName: this.text ? this.text : this.selectList[index].id
            }
            this.currentGw = this.wgList[index]
            this.gwInfo = await this.getDeviceByGwId(gateWayInfo)
            this.comList = this.gwInfo.comRtus
            this.com = Object.keys(this.comList).map(Number)
          })
        Stack({ alignContent: Alignment.Center }) {
          Image($r('app.media.cloud')).width(120)
          Text('云服务').margin({
            top: 20,
            left:-10
          })
        }
        Line()
          .width(100)
          .height(50)
          .startPoint([50, 0])
          .endPoint([50, 50])
          .stroke(0x4F94FE)
        Column() {
          Text('名称:' + this.gwInfo.name).fontSize(15)
          Text('ID:' + this.gwInfo.id).fontSize(15)
          if (!this.gwInfo.status) {
            Row() {
              Text('状态:正常').fontSize(15)
              Circle({ width: 15, height: 15 }).fill(0x25E200)
            }.justifyContent(FlexAlign.Center)
          } else {
            Row() {
              Text('状态:中断').fontSize(15)
              Circle({ width: 15, height: 15 }).fill(0xFF4734)
            }.justifyContent(FlexAlign.Center)
          }
        }
        .backgroundColor(0xc7ddff)
        .padding(10)
        .border({
          width: 3,
          color: 0x4F94FE,
          radius: 10,
```

```
            style: BorderStyle.Solid
          })
          .alignItems(HorizontalAlign.Start)
          .justifyContent(FlexAlign.Center)
          .width(250)
          Line()
            .width(250)
            .height(50)
            .startPoint([125, 0])
            .endPoint([125, 50])
            .stroke(0x4F94FE)
          ForEach(this.com, (item: string) => {
            Column() {
              Text(`COM${item}`)
            }
            .border({
              width: 7,
              color: 0x4F94FE,
              radius: 15,
              style: BorderStyle.Solid
            })
            .padding(10)
            .width(100)
            Line()
              .width(100)
              .height(50)
              .startPoint([50, 0])
              .endPoint([50, 50])
              .stroke(0x4F94FE)
            Flex({ wrap: FlexWrap.Wrap, justifyContent: FlexAlign.Center,
alignItems: ItemAlign.Center }) {
              ForEach(this.comList[item], (i: DeviceInfo) => {
                Column() {
                  Text('名称:' + i.name).fontSize(10)
                  Text('ID:' + i.id).fontSize(10)
                  if (!i.status) {
                    Row() {
                      Text('状态:正常').fontSize(10)
                      Circle({ width: 10, height: 10 }).fill(0x25E200)
                    }
                  } else {
                    Row() {
                      Text('状态:中断').fontSize(10)
                      Circle({ width: 10, height: 10 }).fill(0xFF4734)
                    }
                  }
                }
                .border({
                  width: 5,
                  color: 0x4F94FE,
                  radius: 15,
                  style: BorderStyle.Solid
                })
                .padding(5)
                .alignItems(HorizontalAlign.Start)
                .width(160)
                .margin({
                  top: 5,
                  left: 5
                })
              }, (item: GateWay) => item.id)
            }
            .padding({
              left: 10,
```

```
            right: 10,
            top: 30,
            bottom: 30
          })
          .border({
            width: 1,
            color: 0x4F94FE,
            radius: 15,
            style: BorderStyle.Dashed
          })
          .width('95%')
          .margin({
            bottom: 20
          })
        }, (item: number) => item.toString())
      }
    } .width('100%').align(Alignment.Top)
  }
}
@Entry
@Component
struct Log {
  @State message: string = 'Hello World';
  @State logList: LogInfo[] = []
  private selectedDate: Date = new Date()
  @State endTime: string = ''
  @State startTime: string = ''
  @State searchText: string = ''
  pageSize: number = 20
  page: number = 1
  hasMore:boolean = false
  async aboutToAppear() {
    let queryParameter: LogQueryData = {
      size: this.pageSize,
      current: this.page,
      etime: this.formatDate(new Date()) + " 23:59:59",
      stime: this.formatDate(new Date()) + " 00:00:00"
    }
    this.startTime = this.formatDate(new Date())
    this.endTime = this.formatDate(new Date())
    let data =await this.getLogList(queryParameter)
    this.logList = data.records
    if(this.logList.length==data.size){
      this.page++
      this.hasMore = true
    }else {
      this.hasMore = false
    }
  }
  updateLogging(queryParameter: LogQueryData){
    console.log(JSON.stringify(queryParameter))
    if(this.hasMore){
      this.getLogList(queryParameter).then((result) => {
        let data = result.records
        if (data.length === result.size) {
          this.page++
          this.hasMore = true
        }else {
          this.hasMore = false
        }
        this.logList = this.logList.concat(data);
      }).catch((err: string | Resource) => {
      })
    }else {
```

```
      AlertDialog.show(
        {
          title: '提示',
          message: '已没有更多数据',
          autoCancel: true,
          alignment: DialogAlignment.Center,
          confirm: {
            value: '确定',
            action: () => {
            }
          },
          cancel: () => {
          },
        })
    }
  }
  formatDate(date: Date): string {
    return `${date.getFullYear()}-${String(date.getMonth() + 1).padStart(2,
'0')}-${String(date.getDate() )
      .padStart(2, '0')}`;
  }
  getLogList(queryParameter: LogQueryData): Promise<LogResult> {
    return new Promise(async (resolve: Function, reject: Function) => {
      httpRequestGet(Constants.SERVER +
`/blade-log/api/page?size=${queryParameter.size}&current=${queryParameter.c
urrent}`,{etime:queryParameter.etime,stime:queryParameter.stime})
        .then((result: ResponseResult) => {
          if (result && result.code === 200) {
            resolve(result.data);
          } else {
            reject('wrong');
          }
        })
        .catch((err: Error) => {
          reject('error');
        });
    });
  }
  build() {
    Column() {
      Row() {
        TextInput({
          placeholder: '输入查询',
        })
          .width('28%')
          .textAlign(TextAlign.Center)
          .onChange(async (value: string) => {
            this.searchText = value
          })
      }.width('100%')
      .padding({
        top: 10,
        left: 10
      })
      Row() {
        Text('开始时间')
        Button(this.startTime)
          .backgroundColor(0xf3f3f3)
          .fontColor(0x727272)
          .onClick(() => {
            CalendarPickerDialog.show({
              selected: this.selectedDate,
              onAccept: (value) => {
                this.startTime = this.formatDate(value)
```

```
          }
        })
      }).margin({
      left: 10
    })
  }.width('100%').padding({
    top: 10,
    left: 10
  })
  Row() {
    Text('结束时间')
    Button(this.endTime)
      .backgroundColor(0xf3f3f3)
      .fontColor(0x727272)
      .onClick(() => {
        CalendarPickerDialog.show({
          selected: this.selectedDate,
          onAccept: (value) => {
            this.endTime = this.formatDate(value)
          }
        })
      })
      .margin({
        left: 10
      })
  }.width('100%').padding({
    top: 10,
    left: 10
  })
  Row() {
    Button('搜索').onClick(async () => {
      this.page=1
      let queryParameter: LogQueryData = {
        size: this.pageSize,
        current: this.page,
        etime: this.endTime + " 23:59:59",
        stime: this.startTime + " 00:00:00",
        title: this.searchText
      }
      let data =await this.getLogList(queryParameter)
      this.logList = data.records
      if(this.logList.length==data.size){
        this.page++
        this.hasMore = true
      }else {
        this.hasMore = false
      }
    })
  }.width('100%').padding({
    top: 10,
    left: 10
  })
  Row() {
    Text('时间').width('30%').textAlign(TextAlign.Center)
    Text('操作者').width('20%').textAlign(TextAlign.Center)
    Text('操作行为').width('50%').textAlign(TextAlign.Center)
  }.width('100%').padding({
    right:20
  })
  .height(40)
  .alignItems(VerticalAlign.Center)
  .backgroundColor(0xFFFFFF)
  Scroll(){
    Column() {
```

```
          ForEach(this.logList, (item: LogInfo) => {
            Row() {
              Text(item.createTime).width('30%').textAlign(TextAlign.Center)
              Text(item.createBy).width('20%').textAlign(TextAlign.Center)

Text(item.title).width('50%').textAlign(TextAlign.Center) .fontFamily('Harmo
nyHeiTi-Bold')
                .fontSize(21)
                .fontColor('#182431')
                .fontWeight(700)
                .maxLines(2)
                .textOverflow({ overflow: TextOverflow.Ellipsis })
            }.height(50).padding({
              right:20
            })
            Divider().strokeWidth(3).color('#F1F3F5')
          }, (item: LogInfo) => item.id)
        }.margin({
          top: 5
        })
      }.layoutWeight(1).align(Alignment.Top).onScrollEdge((side: Edge) => {
        if (side == Edge.Bottom) {
          let queryParameter: LogQueryData = {
            size: this.pageSize,
            current: this.page,
            etime: this.endTime + " 23:59:59",
            stime: this.startTime + " 00:00:00",
            title: this.searchText
          }
          this.updateLogging(queryParameter)
        }
      })
    }
    .height('100%')
    .width('100%')
  }
}
```