云开发实践

闹钟日历小程序

目录

1 项目概述

这是一个日程管理工具，闹钟日历提醒。

在此项目实战中，你将通过微信小程序订阅消息云调用能力和定时触发能力，轻松实现待办事项定时提醒功能。通过本项目的简略开发，带你领略小程序·云开发的能力，提升功能开发效率。

2 任务目标

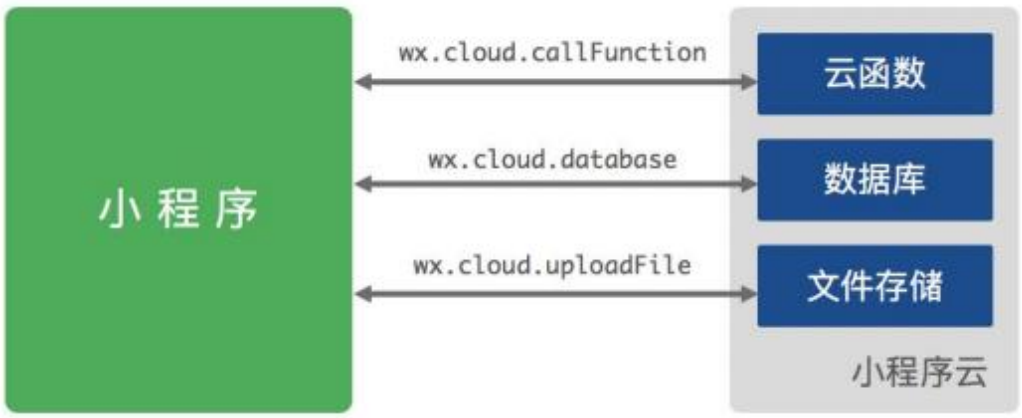
根据项目初始框架，完善代码；使得待办事项定时提醒功能能够正常使用，完成待办事项小程序的开发，并预览至手机进行小程序的体验。

3 准备工作

打开微信小程序开发者工具，选择【闹钟日历-待办事项提醒】项目，等待项目加载完成。

如果有任何异常，请联系周边工作人员寻求帮助。

4 实战架构



5 实战任务

5.1 修复待办事项上传功能

1、打开 miniprogram/pages/index/index.js 文件，找到【代码观看位置 A-1】。此部分为代码理解，无需写代码。这里的函数作用为“申请订阅消息”功能；当用户点击“同意获取订阅消息 “时，函数会将订阅消息发送所需要的信息传入数据库中进行保存。在这里使用了云函数 subscribe 执行数据库保存的操作，代码观看 A-1 展示的是云函数的调用过

```
wx.requestSubscribeMessage({
  // 传入订阅消息的模板id，模板 id 可在小程序管理后台申请
  tmplIds: [lessonTmplId],
  success(res) {
    // 申请订阅成功
    if (res.errMsg === 'requestSubscribeMessage:ok') {
      wx.showLoading({
        title: '订阅中',
        mask:true
      })
      /*【开始】-----------------【代码观看位置A-1】-----------------【开始】*/
      wx.cloud.callFunction({
        name: 'subscribe',
        data: {
          data: item,
          date: date,
          templateId: lessonTmplId,
        },
      }).then(() => {
        wx.hideLoading();
        wx.showToast({
          title: '订阅成功'
        });
      }).catch(() => {
        wx.showToast({
          title: '订阅失败',
          icon: 'none'
        });
      });
      /*【结束】-----------------【代码观看位置A-1】-----------------【结束】*/
    }
  },
});
```

程，以及云函数执行完毕后执行 then 和 catch。

2、打开 cloudfunctions/subscribe/index.js，定位到【代码实战位置 B-1】，请将红框代码填写到【代码实战位置 B-1】开始 和结束之间，使整个函数流程完整。

空缺的代码执行的是数据库添加数据操作。

```
exports.main = async (event, context) => {
  try {
    /*【开始】-----------------【代码实战位置B-1】-----------------【开始】*/
    const result = await db.collection('messages').add({
      data: {
        touser: event.userInfo.openId,
        page: 'index',
        data: event.data,
        templateId: event.templateId,
        date: new Date(event.date),
        done: false,
      },
    });
    return result;
    /*【结束】-----------------【代码实战位置B-1】-----------------【结束】*/
  } catch (err) {
    console.log(err);
    return err;
  }
};
```

*****右键文件列表 subscribe 文件夹，选择【上传并部署：云端安装依赖】上传。**

5.2 修复订阅消息发送功能

1.在申请订阅消息权限后，需要按照业务要求，自行执行订阅消息发送过程。云开发提供订阅消息的云调用能力，开发者能够轻松完成订阅消息的发送。

2.打开 cloudfunctions/send/index.js，定位到【代码实战位置 C-1】，将红框中的代码填写到【代码实战位置 C-1】开始和结束之间，使整个函数流程完整。

此处代码根据数据库中取出匹配待发的数据，循环执行订阅消息发送过程。

```
console.log(messages);
// 循环消息列表
/*【开始】-----------------【代码实战位置C-1】-----------------【开始】*/
const sendPromises = messages.data.map(async message => {
  try {
    // 发送订阅消息
    await cloud.openapi.subscribeMessage.send({
      touser: message.touser,
      page: message.page,
      data: message.data,
      templateId: message.templateId,
    });
    // 发送成功后将消息的状态改为已发送
    return db.collection('messages').doc(message._id).update({
        data: {
          done: true,
        },
      });
  } catch (e) {
    return e;
  }
});
/*【结束】-----------------【代码实战位置C-1】-----------------【结束】*/
return Promise.all(sendPromises);
```

*****右键文件列表 send 文件夹，选择【上传并部署：云端安装依赖】上传。**

5.3 增加订阅消息定时发送功能

1.在业务需求中，我们需要一个特定的时间执行订阅消息的发送，所以我们需要让 send

云函数能够周期性触发。来实现特定时间点发送订阅消息的功能。

2.打开 cloudfunctions/send/config.json，里面含有如下内容：

```
{
  "permissions": {
    "openapi": [
      "subscribeMessage.send"
    ]
  }
}
```

此为云调用的配置项，在执行订阅消息发送之前，需要在这里声明。

3.在此文件中编辑，使之变成如下。此为云函数的定时触发器配置，规定每 5 秒触发 1 次，具体关于触发器内容请扫码查看。

```
{
  "permissions": {
    "openapi": [
      "subscribeMessage.send"
    ]
  },
  "triggers": [
    {
      "name": "sendMessagerTimer",
      "type": "timer",
      "config": "1/5 * * * * * *"
    }
  ]
}
```

*****右键文件列表 send 文件夹，选择【上传触发器】进行触发器部署。**

6 完成标准

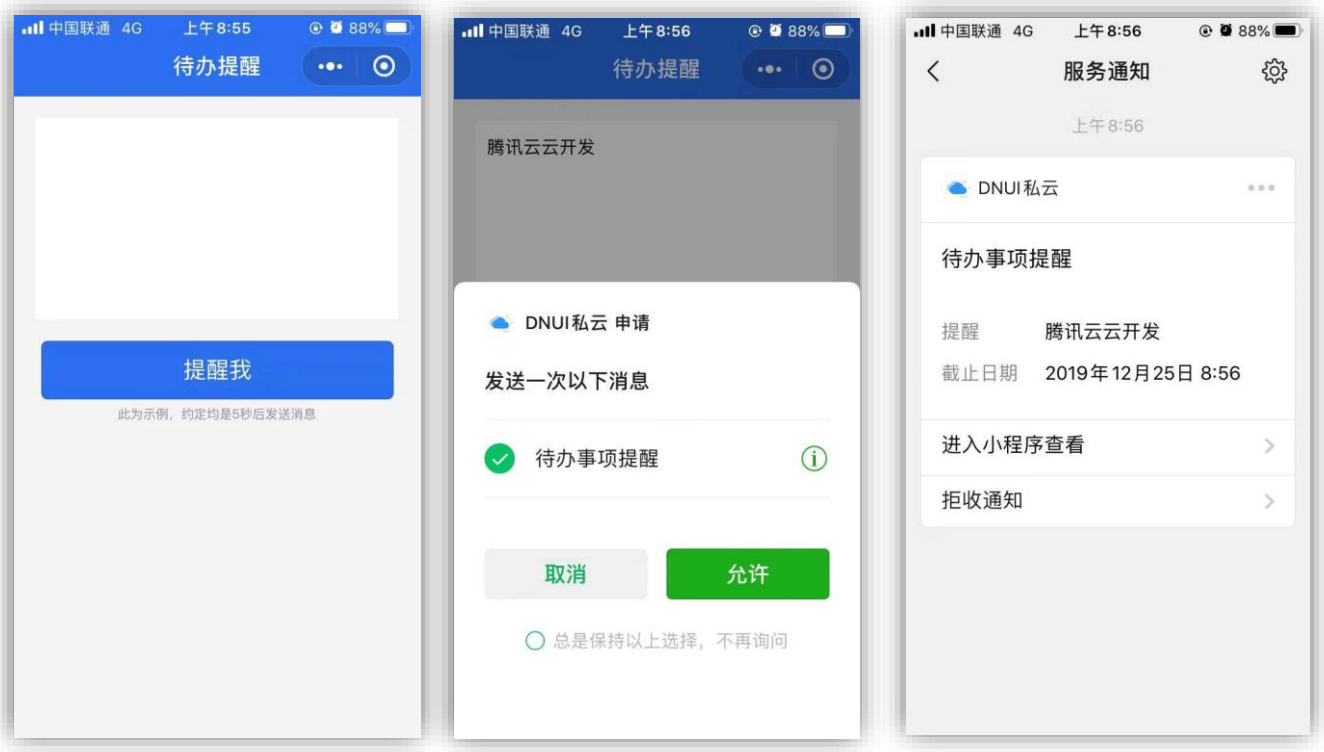
1、待办事项提醒小程序的功能都可以正常使用。

2、待办事项数据使用云数据库能力存储在云端，并从云端拉取展示。

3、手机服务消息可以收到小程序发送的待办提醒。

4、任何页面没有 Warning 报错。

7 结果页面



关注腾讯云「云开发」

上小程序，用云开发