授权组件 login

* 判断用户是否授权(在onpublish中)
* 如果scope.userInfo为true，则代表用户被授权过
* 如果授权，getUserInfo获取用户的对应信息。如果没授权，就弹出底部弹出层
* Bottom modal里写个slot
* 建立一个login组件，为bottom-modal的父组件， 里面写个slot, 名字为bottom-modal的modal-content
* 在blog page中调用login, 并把modalshow变量传过去
* 在login和bottom-modal中分别接受传过来的变量
* 在login中的按钮里，open-type设为getUserInfo, 并绑定点击事件
* 在点击事件中，拿到事件的userInfo. 如果有userInfo, 代表拿到了用户信息，关闭下显示框(modalshow为false)，并抛出拿到的方法和用户信息一起到blog page。 如果没拿到，抛出没拿到的方法到blog page
* 如果用户已经授权，那么就跳转到发布界面，并把用户昵称和头像传过去
* 再到onpublish的getuserinfo中调用之前抛出的成功的方法

发布页面

* 对blog-edit进行布局，包含整个container和footer，footer中包含记录当前评论字数的变量
* Textarea为原生组件(原生组件的层级不能被z-index修改，原生组件也不能显示在容器里面，比如scroll-view等等，不能css绝对定位)， 用bindinput来绑定输入事件（注意不能bind:input, 写了冒号就绑定不上了），在里面拿到用户输入的value
* 如果用户输入的值大于140，那么输出提示最大字数为140(在textarea里定义maxlength来获取)
* 输入的时候，键盘会弹出来，底部的footer会被移到上面，键盘收掉后footer又会回到原位

1. 给textarea 添加auto-focus， 再绑定bindfocus, bindblur函数
2. 把footer的style改成动态的，bottom的值改为一个变量footerbottom
3. 在textarea的两个绑定函数中，其中在focus的时候把footerbottom设为当前事件中的高度， 在blur的时候把footerbottom设为0

选择图片

* 设一个最大图片选择数量max\_img为 9， 再设一个数组images来存放图片，他们俩的差就是现在还能再添加的图片，把这个数放到chooseimage的count里
* 绑定选择图片事件，并在里面写wx.chooseimage，里面设count, sizetype[original, compressed], sourcetype:[album, camera], 成功之后的结果里面的tempfilepath应该追加到当前数组中
* 在wxml页面中循环显示，key就是\*this， 并把图片模式设置为aspectFill。
* **当传了9张图片时，把那个加号图片隐藏**：

1. 之前的selectphoto样式绑定boolean控制hidden, 默认为true, 当max\_img与当前images数组长度的差为0，就代表9张选满了，selectphoto就变成false，反之亦然

* **图片删除:**

1. 给每一张图片添加删除矢量图，绑定删除事件，再传入data-index为当前index(确定当前绑定的是第几张)。
2. 在绑定事件中传入event, 在images数组中用splice方法把当前这个index给去掉，在把images数组重新setdata为新的images
3. 如果images数组长度比max\_img少1， 那就重新显示带加号的图片(selectphoto再设为true)

* **PreView 图片：**

1. 给图片绑定preview事件，再传个自定义属性过去(为当前的item)
2. 然后事件中的url就是这个images数组， current就是传过来的事件中的属性

**文件上传云存储**

要传到数据库的：评论内容，图片fileID, openid, 昵称，头像，时间

图片上传

* 给发布按钮绑定事件
* 循环遍历图片数组，把每一张图片放入云数据库
* 在wx.cloud.uploadFile中，cloudPath, filePath
* 正则表达式选取图片的后缀格式

存入到云数据库

* showloading

把每一次云上传都作为一个promise对象，建个数组来存放这些对象

* PromiseAll 所有的promise数组中的对象。 之后从云数据库中的collection里分别加入评论内容，图片fileID, openid, 昵称，头像，时间

1. 如果评论为空，弹窗提示，并直接return
2. 设一个空数组，把图片的res.fileid合并到此数组中，并传进数据中
3. 把所有userinfo传进来
4. 把时间传进来

* 全存完之后hideloading
* 再navigateBack()回发现的界面

发布内容卡片:

* 去后端建一个新的云函数，安装tcb-router
* 建立一个list路由，skip从start开始，limit数量为count， orderBy时间为desc(逆序), 然后get。 把这一串赋予一个值，把这个值赋给ctx.body
* 去到find page里，获取云函数，建立一个空数组，并把获取函数后的res.result追加到此数组里
* 封装卡片组件，并渲染显示内容
* 单独写一个格式化时间的js文件，并在find-card(卡片组件)中引用

优化内容卡片:

* 在调用find云函数之前跳出加载窗口，之后关闭窗口，设置默认的start 初始值为0。触底的时候把start的初始值变成当前bloglist的长度。上滑刷新的时候再次调用获取云函数的方法，初始值再为0，并且清空bloglist数组, 再去json文件中enablePullDownRefresh为true。之后再到方法中stopPullDownRefresh()
* 给每张图片绑定previewimage事件(注意要用catch:tap绑定，这样能防止事件冒泡)
* 建立blogComment页面，给每个博客卡片绑定点击事件，使其进入blogComment
* 每次新添加一个blog, 都要刷新整个find页面并显示新的blog

1. 在blog-page的naviagteBack后面，用getCurrentPages拿到page数组，并拿到其长度之前两个的索引。再用这个调用父组件find里面的pulldownrefresh生命周期函数

**模糊搜索:**

* 在搜索组件中分别给搜索框和按钮绑定事件，在按钮点击后拿到keyword，并抛出事件和keyword到父组件(find)
* 在父组件接受抛出的事件，事件中通过event拿到keyword的值， 把bloglist重置为空，并再次调用获取blog的方法，默认从0开始, 再把keyword传入调用云函数的方法中
* 去到blog云函数中，在list路由里，通过event拿到keyword。设个find对象默认为空，再判断keyword是否为空。如果不为空，把find对象通过对keyword正则表达的处理方法 content: new db.RegExp({regexp:keyword, options:’I’})(这就代表不看大小写的模糊搜索)
* 之后在下面的await语句中加where(find)， 这样如果find为空， 那where就什么也没做，如果不为空，就搜索输入的keyword

提高查找效率： 云数据库中创建索引(空间换时间)

* 索引名称可以自己定义，索引字段必须和数据库里一样，比如我们用content做为索引字段，那我们搜索的content效率就会提高