多功能日历ValCalendar 实训结题报告

软件2班 杨璐玮 202230483091

[题目要求/项目简介 1](#_Toc139005934)

[项目架构与业务逻辑设计 2](#_Toc139005935)

[解决方案文件架构 2](#_Toc139005936)

[业务逻辑设计 3](#_Toc139005937)

[核心功能及其实现 4](#_Toc139005938)

[用户登陆注册 4](#_Toc139005939)

[主界面 5](#_Toc139005940)

[日程实时增删改查可视化 7](#_Toc139005941)

[团队表以及日程动态可视化 10](#_Toc139005942)

[数据永久保存——文件读写操作（以登录注册和团队创建模块为例）： 14](#_Toc139005943)

[基于日程类型和紧急程度的出发时间推荐算法 16](#_Toc139005944)

[软件性能优化——地图热启动 17](#_Toc139005945)

[心得总结 18](#_Toc139005946)

[技术选型探索斟酌 18](#_Toc139005947)

[GUI： 18](#_Toc139005948)

[DataBase未遂的尝试： 19](#_Toc139005949)

[API： 19](#_Toc139005950)

[对C++语言设计和软件设计模式的理解： 19](#_Toc139005951)

[项目开发经验总结与反思 19](#_Toc139005952)

[使用Git进行版本管理 19](#_Toc139005953)

[文件读写与软件项目的可维护性 20](#_Toc139005954)

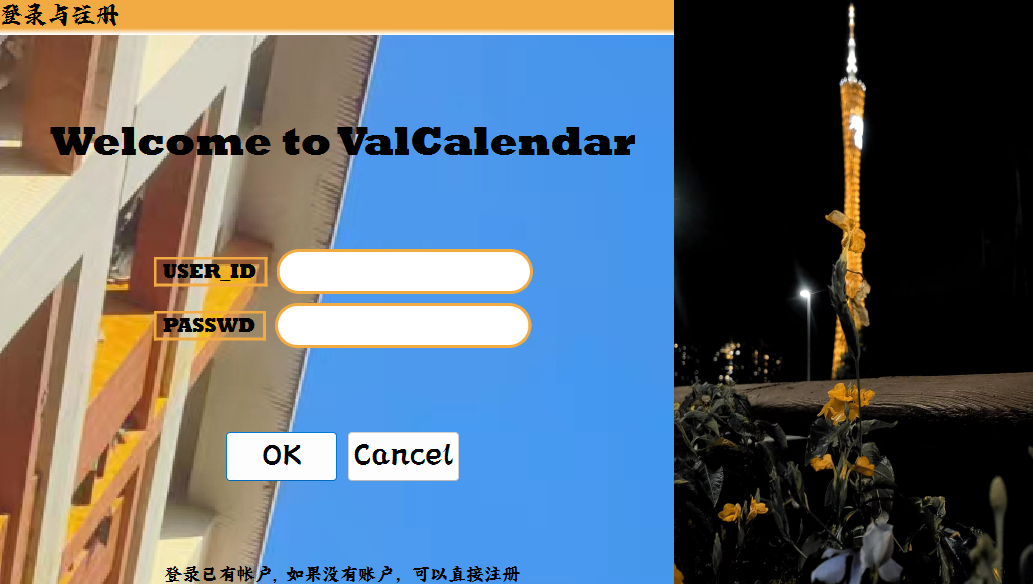
[该项目的拓展可能性 21](#_Toc139005955)

[完善百度地图的业务使用 21](#_Toc139005956)

[客户端和服务端的网络编程实现联网 21](#_Toc139005957)

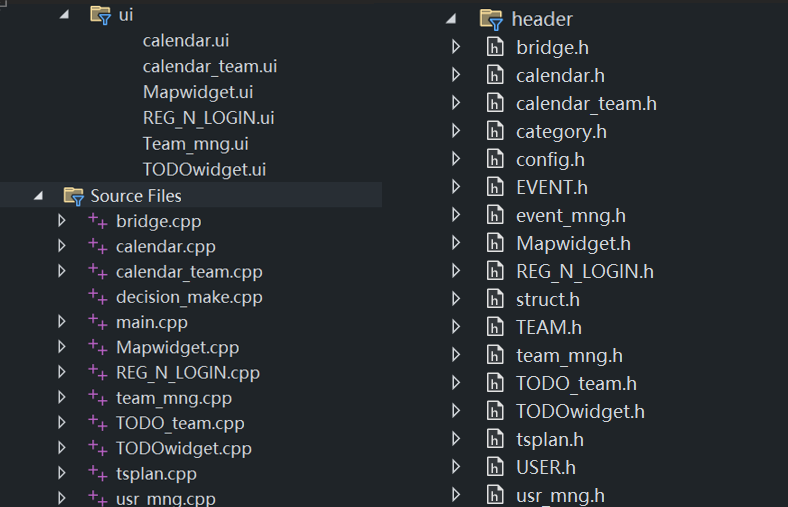
# 题目要求/项目简介

Valcalendar，一个多功能日历软件。在实现日历功能的基础上，实现日程管理功能，包含日程紧急程度等利于团队日程管理的设定；允许多用户、多层次使用，让不同用户都可以创建和管理个人或者团队的日程，并且同步存档。



# 项目架构与业务逻辑设计

## 解决方案文件架构

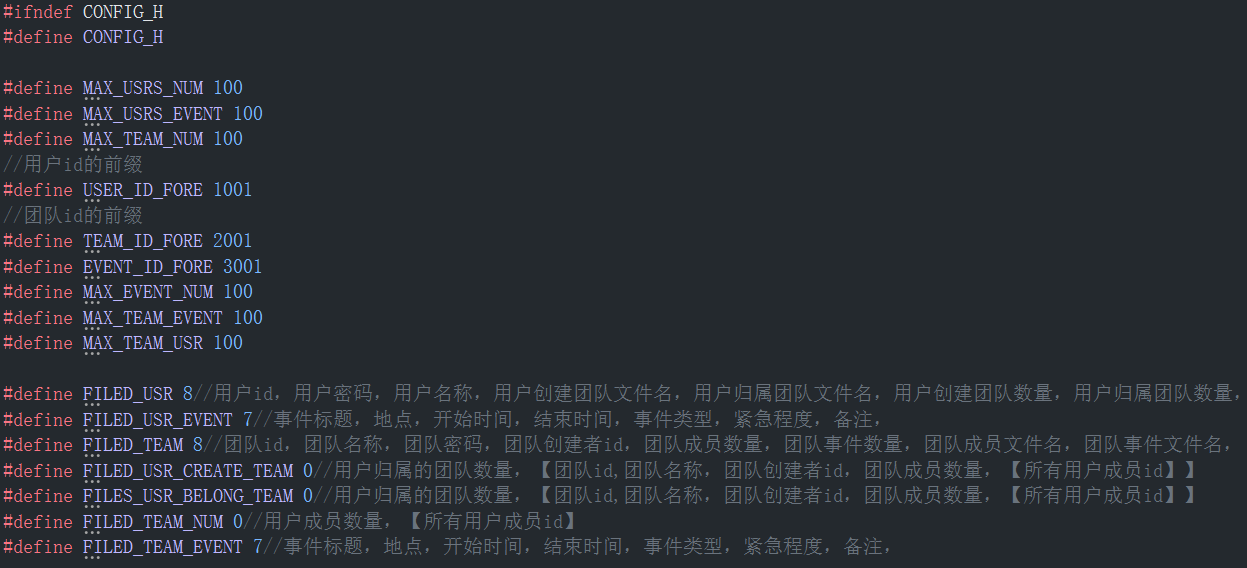


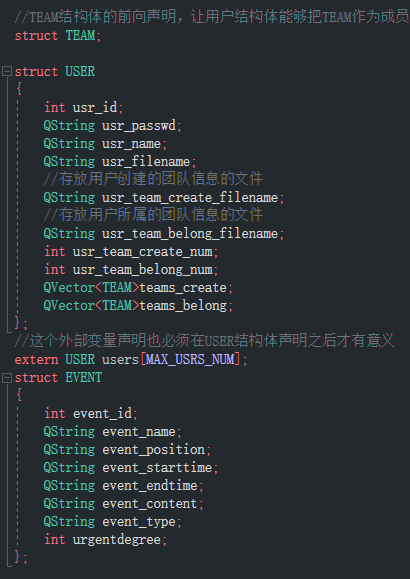
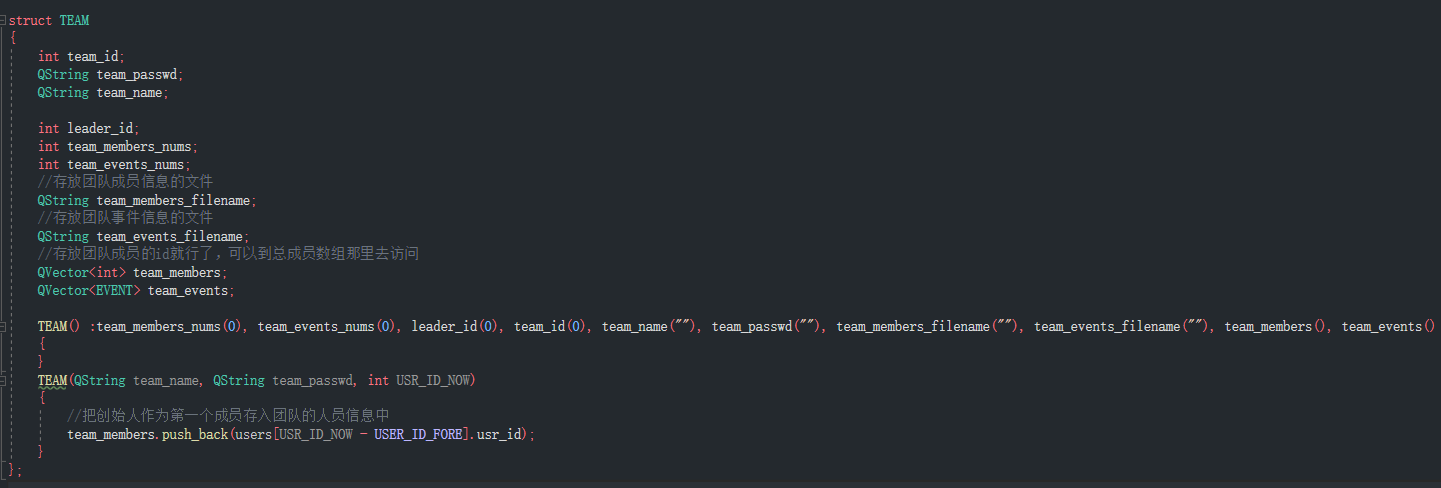
一共定义了六个类，一共有calendar个人主窗口类、calendar\_team团队主窗口类，Mapwidget地图类和REG\_N\_LOGIN，负责登陆注册逻辑的窗口类和TODOwidget日程创建类。其中TODOwidget通过继承重写，在个人日程创建和团队日程创建的功能中实现了代码的复用。

这个项目是基于Qt进行开发的，对象主要就是各类窗口，每个窗口下又包含很多小的控件对象。这些对象都是由Qt提供的许许多多的基类以及他们的派生类实例化得到的。

## 业务逻辑设计

本项目的业务逻辑核心是通过规定一些结构体和文件操作的格式化标准，使用文件操作在用户与软件的各个交互过程中进行实时存档，实现各种数值跟随用户操作改变的变化要求。



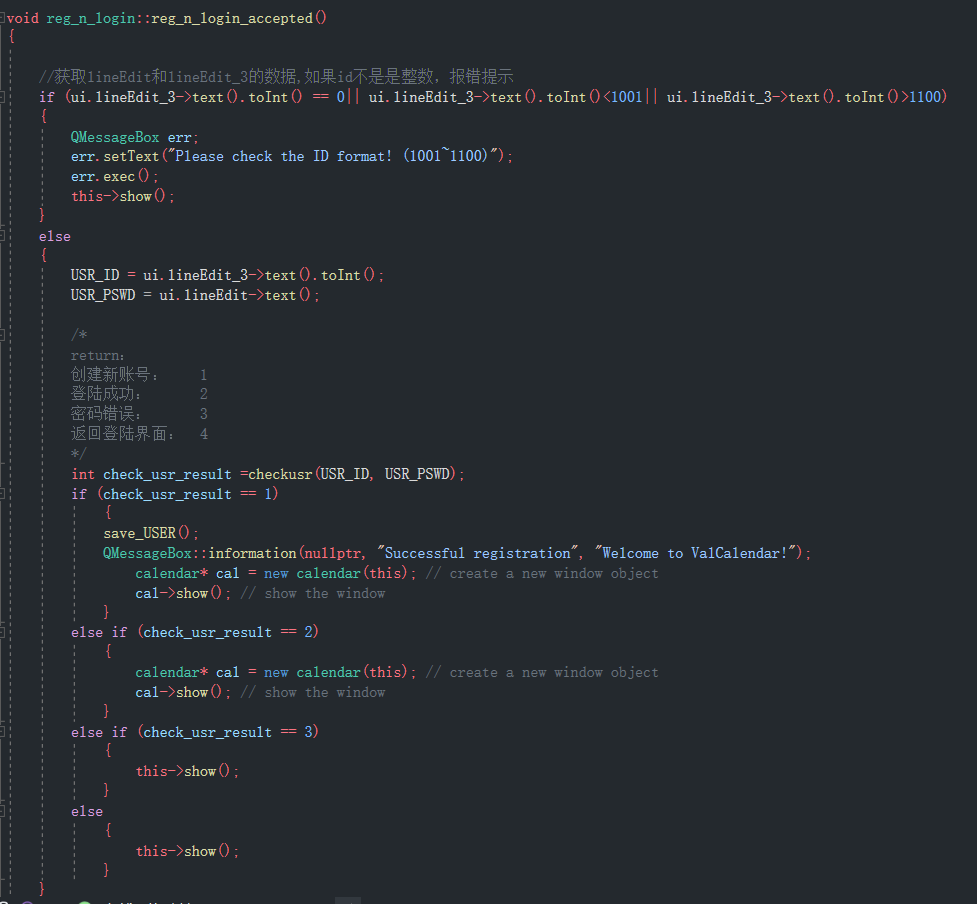
 

# 核心功能及其实现

## 用户登陆注册



打开应用程序，首先是reg\_n\_login窗口。这时候用户输入id和passwd，接下来如果用户点击OK进行登录操作，那么该事件就会connect到reg\_n\_login\_accepted()函数：



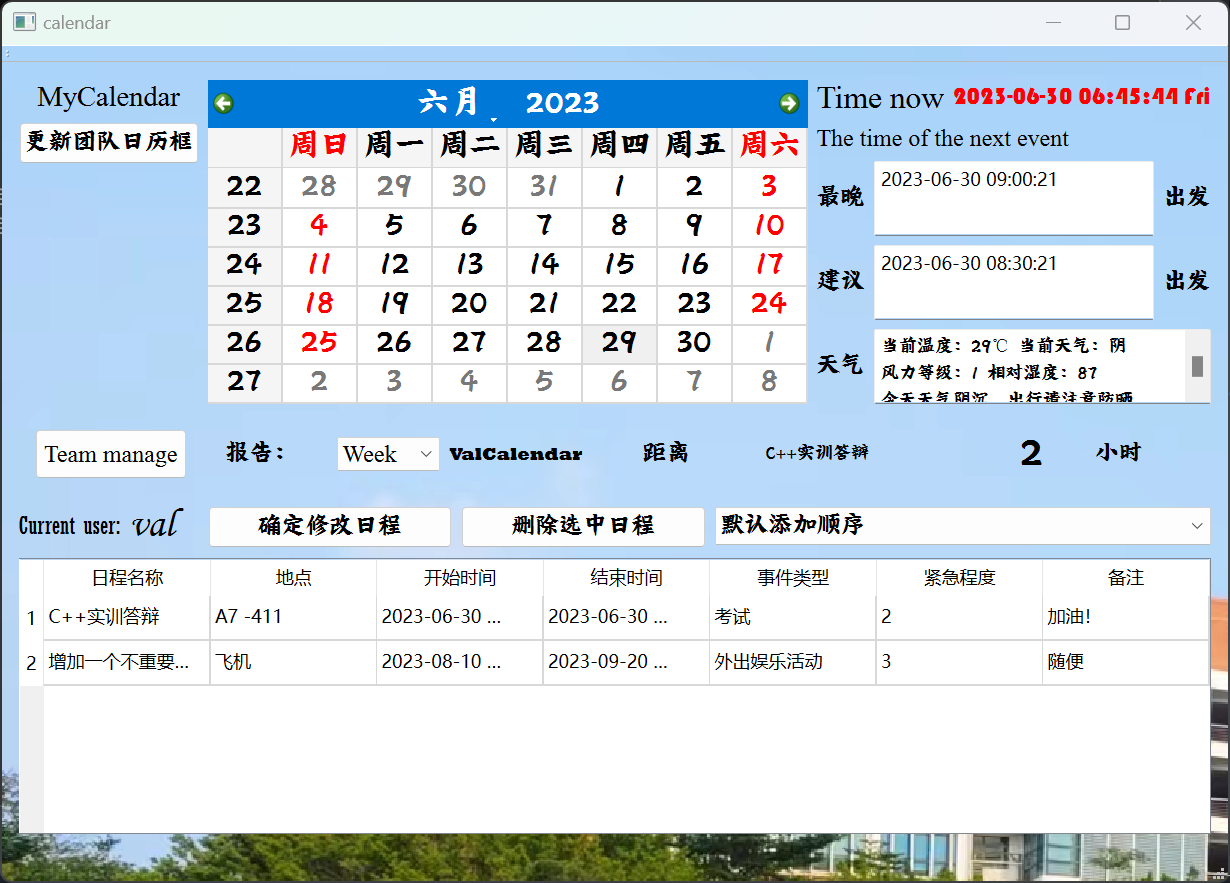
函数还会进行一些合法性判断，根据用户的输入结果导向不同的逻辑。项目使用了QMessageBox类在很多细节的地方进行用户操作的提示和警告。QMessageBox类提供了一些静态函数，用于方便地创建常见类型的消息框，而无需显式创建 QMessageBox 类的实例。

完成注册之后会返回登录窗口并重新初始化用户文件，以便于用户能够继续完成登录。

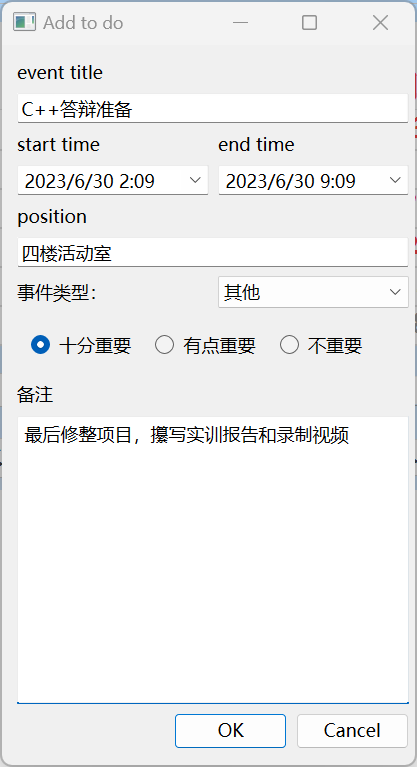
完成登录之后，进入主界面。

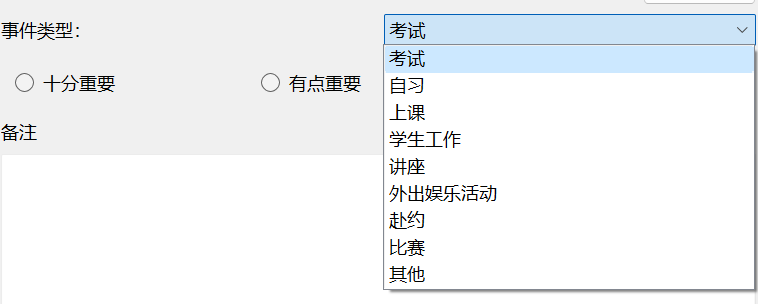
## 主界面

**这个项目将不同用户的账户和日程分开管理。主视图中左处会显示当前用户的昵称。**



## 日程实时增删改查可视化



****

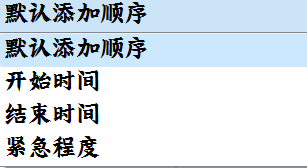
**采用一个下拉框来存放事件类型，并且存储到对应的日程结构体的event\_type中。**



添加好了之后可以在这里看到刚刚添加的日程。

这一步主要是使用QTableWidget实现的。我们从指定的当前用户对应的日程文件夹中读取日程信息之后，指定表格的行列数以及表头，接着分割文段塞入对应的表格里。然后换行，写入下一个日程的信息。

其中用户的信息在登录之后就由全局变量传递到所有的类和变量之中。确保了用户登录账户以及日程、团队操作的统一。



**添加多个日程之后可以选择多种方式进行排序**

**自定义了一个排序函数，根据combobox的值调用不同的排序分支。**

**实现排序功能的过程中遇到问题：**

**QTableWidget: cannot insert an item that is already owned by another QtableWidget**

**使用 clone() 方法是成功解决这个问题的关键，因为它创建了原始单元格对象的副本，而不是直接移动原始对象。**

**克隆副本保留了数据完整性：**

使用 clone() 方法创建原始单元格对象的副本，保留了原始数据的完整性。这样，在重新设置行顺序时，不会直接移动原始对象，从而避免了数据丢失的问题。

**避免了对象所有权混乱：**

单元格对象只能属于一个表格，当尝试将已拥有的单元格对象插入到另一个表格时，会引发错误。通过克隆副本，避免了对象所有权的混乱，确保每个副本单元格都属于正确的目标表格。

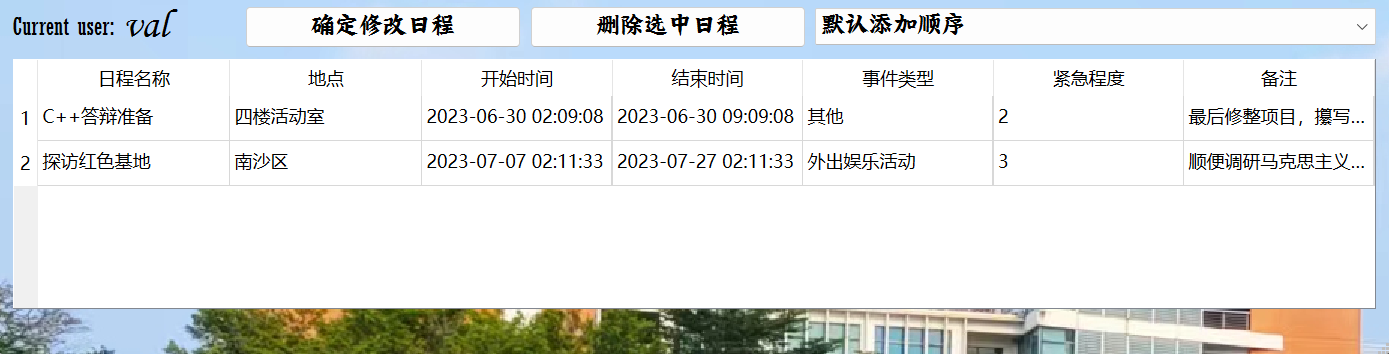
正确处理对象的生命周期：

在使用 clone() 方法创建副本时，需要确保在不再需要副本时及时释放副本对象的内存，以避免内存泄漏。这样可以正确处理对象的生命周期，避免了潜在的内存问题。

总结起来，使用 clone() 方法创建副本是一种常用的解决方案，可以保留原始数据的完整性，并避免对象所有权的混乱。通过正确处理对象的生命周期，可以确保代码的健壮性和可靠性。然而，每个问题都有其独特性，因此在实际应用中，根据具体情况选择适当的方法和技术是非常重要的。

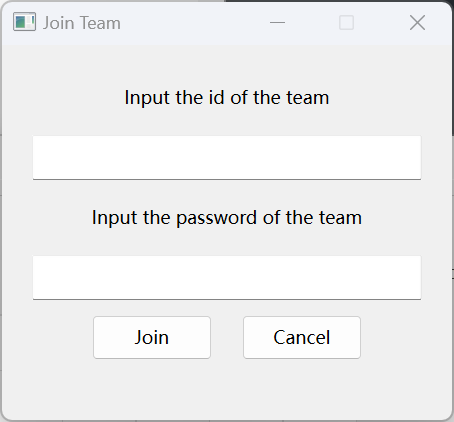
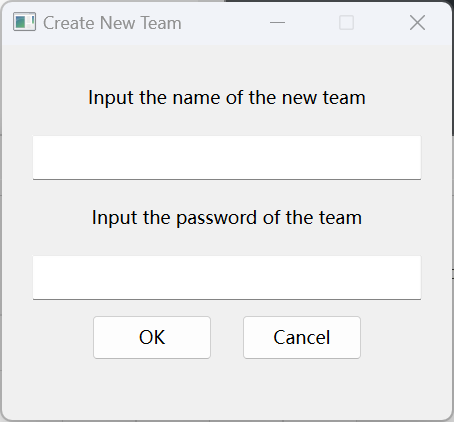
执行修改操作，表格试图中的日程字段会保存这些改变，并且文件的内容也会随之改动。

执行删除操作，表格视图中的第二个日程消失，同时也在文件中删除

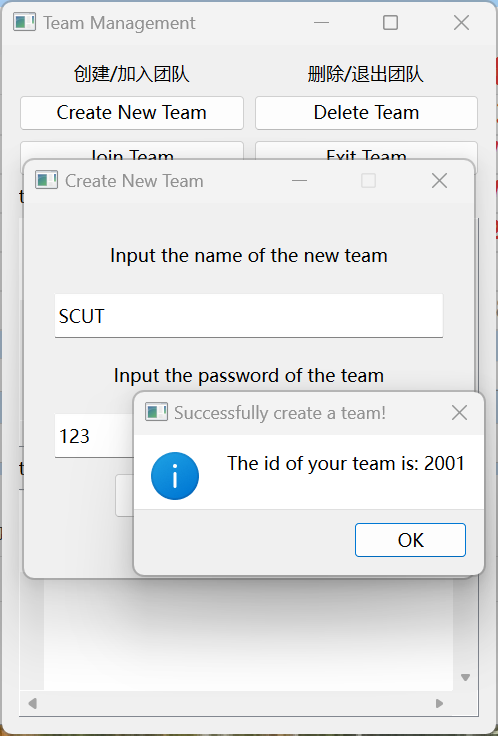


## 团队表以及日程动态可视化

首先需要创建/加入团队

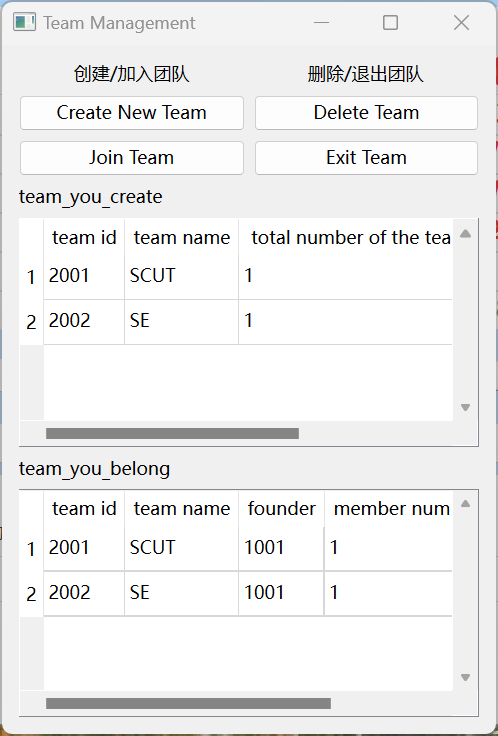


使用team\_id来唯一标识特定的团队，并且在创建/加入的时候需要设置/输入正确密码。



创建团队之后，程序会根据现有的团队数量分配给这个新的团队一个专属的ID。

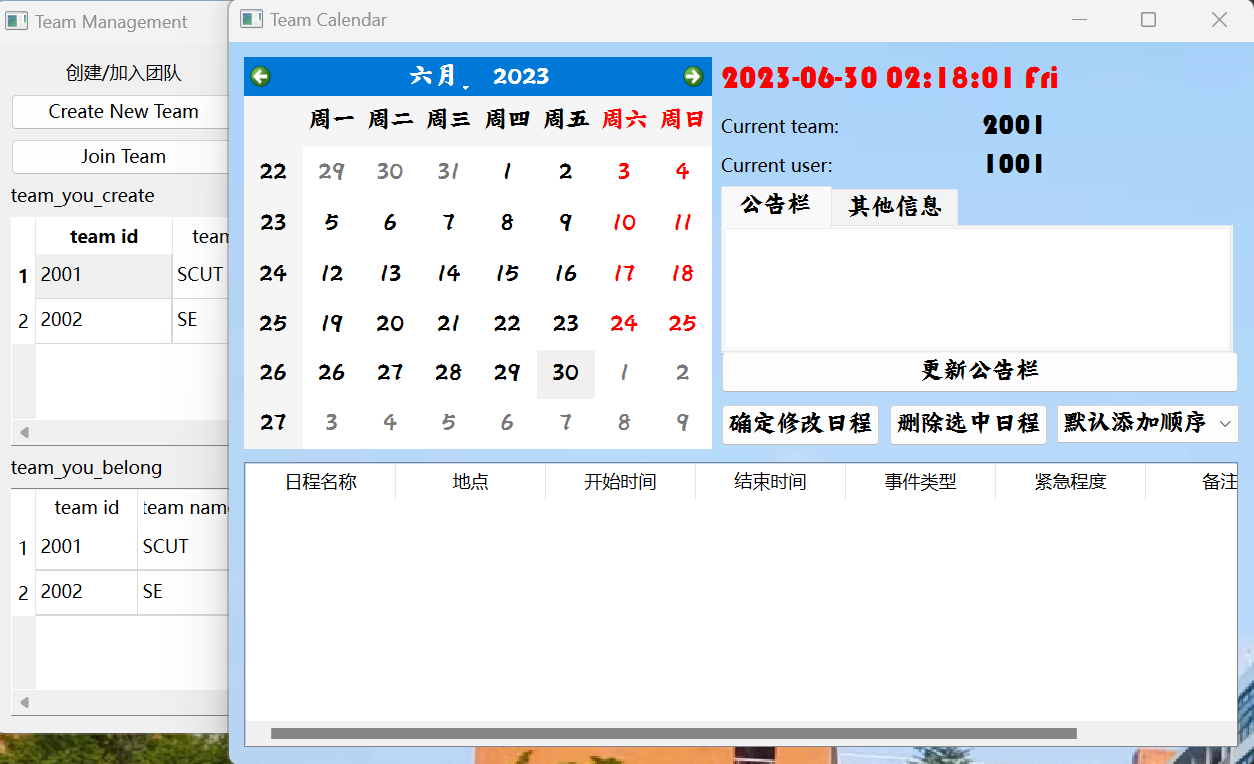
此时团队创建窗口发出新团队创建的信号，被团队管理窗口捕获之后，即刻更新在团队管理窗口的表格视图中。（创建团队的人默认也归属于这个团队）



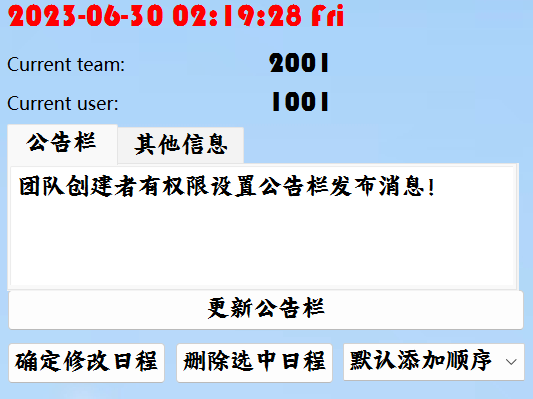
这个时候在主页面的左上角点击更新团队日历框，就会出现新创建的团队。这个功能是通过读取用户归属团队的文件得到团队数目，然后循环创建控件放入布局，依次通过connect函数绑定点击事件和一个往表格视图中新增团队日程的函数。



为了改善用户的体验，我设计了一个双击团队信息视图中的某一行，就能唤起团队日程管理窗口的功能。



设置了一个公告栏，程序检测当前用户的id和该团队的创建者id，来确定和分配对应的权限：只有团队创建者才能更新公告栏和创建发布团队日程。



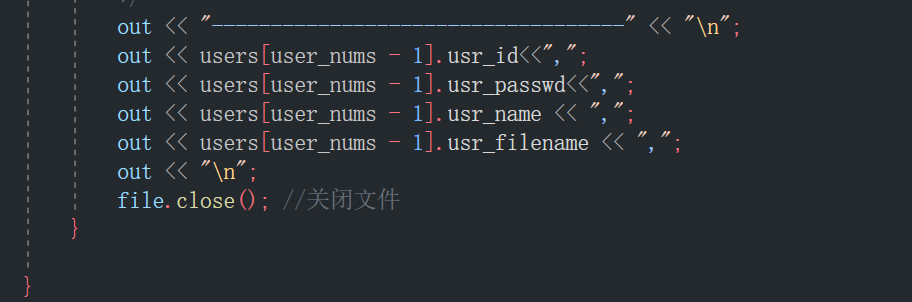


## 数据永久保存——文件读写操作（以登录注册和团队创建模块为例）：

Save\_USER():新用户的写入操作：



User\_nums的值根据用户结构体数组初始化函数从USERS.txt文件中读取的用户数量实时更新，便于新用户的正确写入。



按照用户结构体的顺序，把用户相关的数据字符串以‘,’为分隔符写入USER.txt文件。

用户结构体数组的初始化函数：init\_users\_structarr()

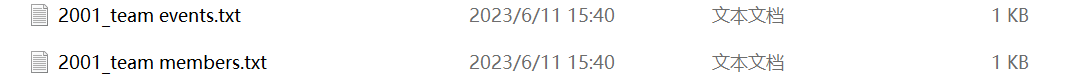


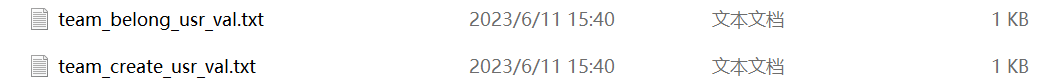
使用split函数以逗号为分隔符切割USERS.txt中的用户数据字段并读入结构体数组中。（后续将根据模块开发进度更新语句）

由此，基本实现了用户登录和注册的流程。

团队创建背后的实现原理是，当用户创建一个团队之后，会依次对以下五个文件进行读写操作，以实现团队信息和用户信息的更新。







## 基于日程类型和紧急程度的出发时间推荐算法

既然是多功能日历，智能推荐是必不可少的。为了设计一个算法来计算建议出发时间和最晚出发时间，需要考虑以下几个因素：

日程的起始时间：根据给定的起始时间，可以直接使用该时间作为建议出发时间。

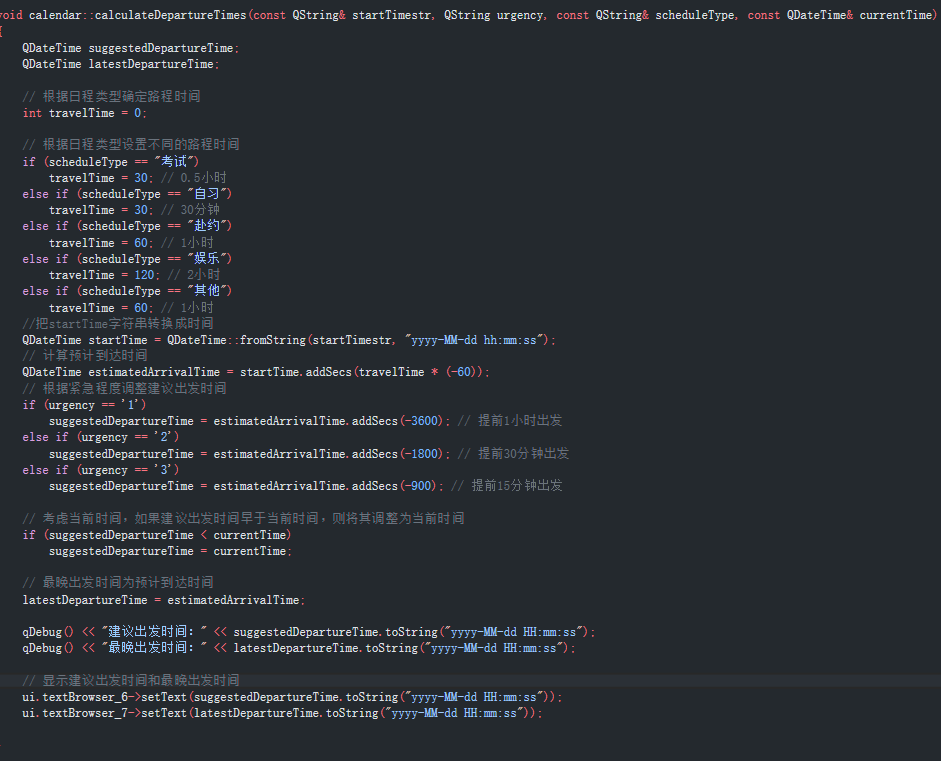
日程类型：对于不同的日程类型，可以预先定义它们的路程时间。根据日程类型和起始时间，可以计算出预计的到达时间。建议出发时间可以是预计到达时间之前的适当时间。

紧急程度：如果日程的紧急程度较高，可以提前出发，以确保按时到达。您可以根据紧急程度调整建议出发时间。

当前时间：根据当前时间，可以确定距离日程开始还有多长时间。根据这个时间，可以进一步调整建议出发时间。

综合考虑这些因素，可以设计一个简单的算法来计算建议出发时间和最晚出发时间。

出发时间由机动时间、通勤时间、当前时间和日程起始时间共同决定。理想状态下通勤时间应该由地图API以及JS脚本给出，比较遗憾的是截止于答辩时间，该功能现在只能辅助用户。那么使用模糊替代法，对于不同的事项类型设置不同的通勤时间，不同的紧急程度又决定了不同的机动时间。



## 软件性能优化——地图热启动

实际上在创建日程的时候，我原本设计过一个功能，通过调用百度地图的API实现用户对于日程地点的便捷选取和调用，就能实现时空的高效日程调度。

（第一版本的地图窗口）

后来就因为各种原因转回了JS API。最终只实现了QT向JS脚本的通信，在获取JS脚本向QT的通信的时候遇到了比较多的问题。只能提供给用户一个地图辅助估计时间的作用了。

但是作为地图辅助，他还存在一个问题，那就是加载时间太长了，用户等不了那么久！

于是我在查阅一些相关资料之后考虑进行性能优化：分离出MapWidget类，作为一个单独的窗口类，在打开日程创建窗口的时候创建对象并且唤起，关闭的时候没有删除，而是“缓存“着。这样做，虽然第一次加载是”冷启动“依然需要较长的时间，但是后面的开启就是”热启动“了，可以做到即点即开。这在一定程度上优化了用户体验。



# 心得总结

## 技术选型探索斟酌

GUI：

easyX和Qt

本项目选择Qt正是因为其拥有丰富的控件，避免了重复造轮子，有利于项目开发更多更有使用价值的功能。

因此，此模块的开发首先在Qt Designer中创建了主程序窗口，其中日历使用的是Qt自带的QCalendarWidget控件。

通过初步对项目需求和功能进行分析，使用布局功能重新设计了主程序窗口。使得内容展现更加规范。

而easyX虽然上手容易，对新手友好，但是比较复杂的功能就很难胜任。

DataBase未遂的尝试：

项目开发之初曾经走了一段弯路，没有意识到不能用数据库，反而是在纠结MySQL和SQLite以及在对他们与QT配合的操作中折腾了很久。最后低并发和低数据量并且面向课程的项目还是需要使用文件操作，以后的其他项目或许这些折腾的经验也能用上。

API：

百度地图，和风天气

由于桌面应用如果想要使用市面上的主流在线地图API，只能通过调用他们的Web API或者JS API来实现。一开始在尝试JS API的时候我吃了经验不足的亏，遇到了比较多的问题，而且网上资料比较散，文档也比较粗略，中间曾短暂地尝试过web api，有了第一版的地图界面，尽管能够显示但是性能并不佳，卡顿和浏览器的阻塞非常明显。

而和风天气这一类简单易用的WEB API对于专项功能的“外包”效率还是非常高的。

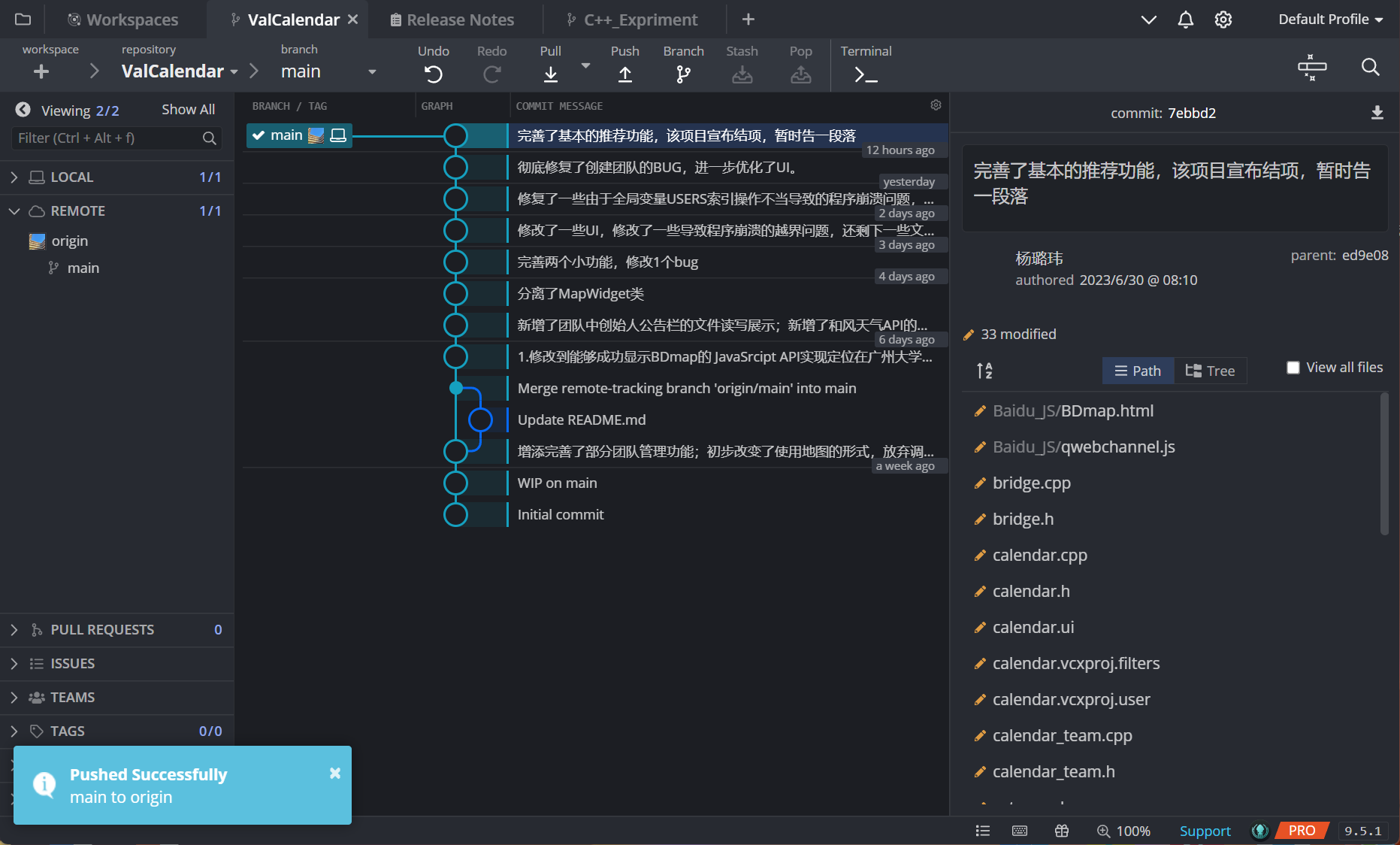
## 对C++语言设计和软件设计模式的理解：

这个项目尤其重要的经验是C++程序设计和软件的设计模式上的反思和理解。一个软件项目如果想开发的更好更易维护更安全，最需要的就是面向对象的代码复用以及类体系架构设计、最小权限原则等思想。纸上得来终觉浅，通过不断地踩坑和折腾，未来语言学习之路和项目开发之路似乎也多了很多明确的导向。

## 项目开发经验总结与反思

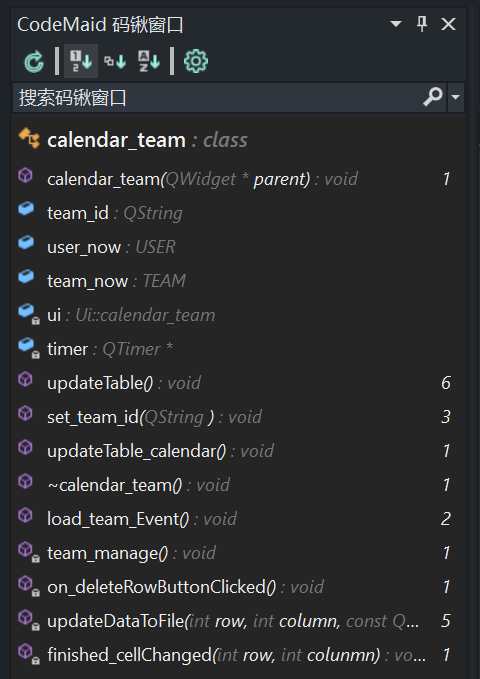
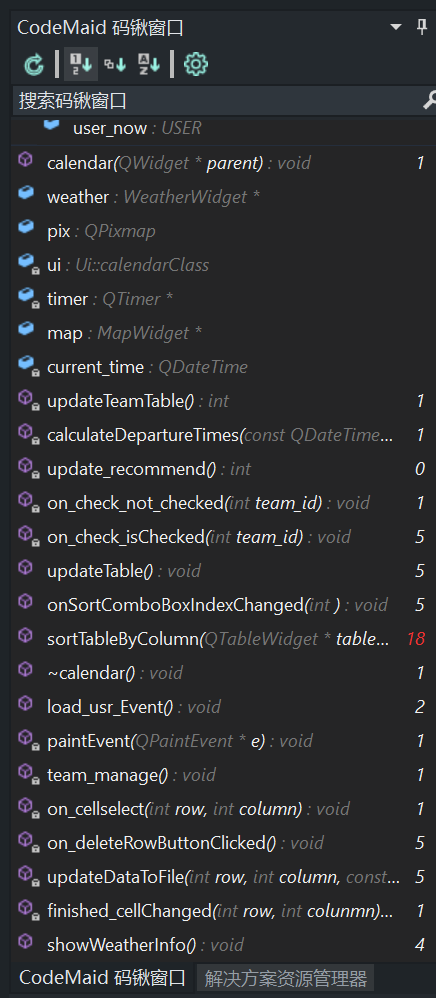
### 使用Git进行版本管理

本次项目开发使用了git工具进行版本控制，我切身感受到了版本控制的优越性，即使只是一个人开发，效率和安全性也提高了不少，git版本控制工具对于项目的长期开发和版本迭代提供了很强大的支持。



### 文件读写与软件项目的可维护性

前期开发开始的时候对文件读写的理解还不够深入，导致项目的文件分布结构设计不太合理，以及文件操作的规范性以及复用率比较低，一定程度上也造成了这个项目存在一些可维护性成本比较高的缺陷。使用码锹窗口分析部分代码的复杂度，有些地方的复用性和可维护性不尽人意，如果有时间重构的话还可以更好。

## 该项目的拓展可能性

### 完善百度地图的业务使用

百度地图的是国内主流的地图API，提供了不少强大的功能。但是由于市面上对于直接的桌面应用软件并没有很直接的接口，导致了跨平台和跨语言地使用存在一些技术政策的门槛。即使我花了很大时间精力钻研，最终的结果也还有很大提升空间。如果能真正实现Qt和JS的双端通信，我相信这个项目能够在生活中发挥不小的作用。

### 客户端和服务端的网络编程实现联网

最开始使用结构体，目的其实还希望最终能够效仿我曾在黑框框程序设计大赛中使用到的socket网络编程开发双端的应用，让用户和用户能真正地实现团队管理和日程调度。可惜的是由于这个项目的逻辑比较复杂，而且受到开发周期的限制和时间分配的错误估计，最终没有时间实现这个想法。