南開大學
Nankai University

计算机学院

高级语言程序设计大作业报告

基于 Qt 的 MyAutoCalendarToDo 应用开发实战

姓名：孟笑朵

学号：2010349

专业：计算机科学与技术

2022 年 5 月 7 日

Abstract

本项目利用 Qt 结合 google Oauth2 验证，借助 google calendar api 接口连接 google calendar，在算法上结合算法导论中的区间调度算法及其拓展问题算法，实现了具有 UI 界面的自动化的个人日程管理软件 MyAutoCalendarToDo。

关键字： Qt, google Oauth2, google calendar api, 区间调度

目录

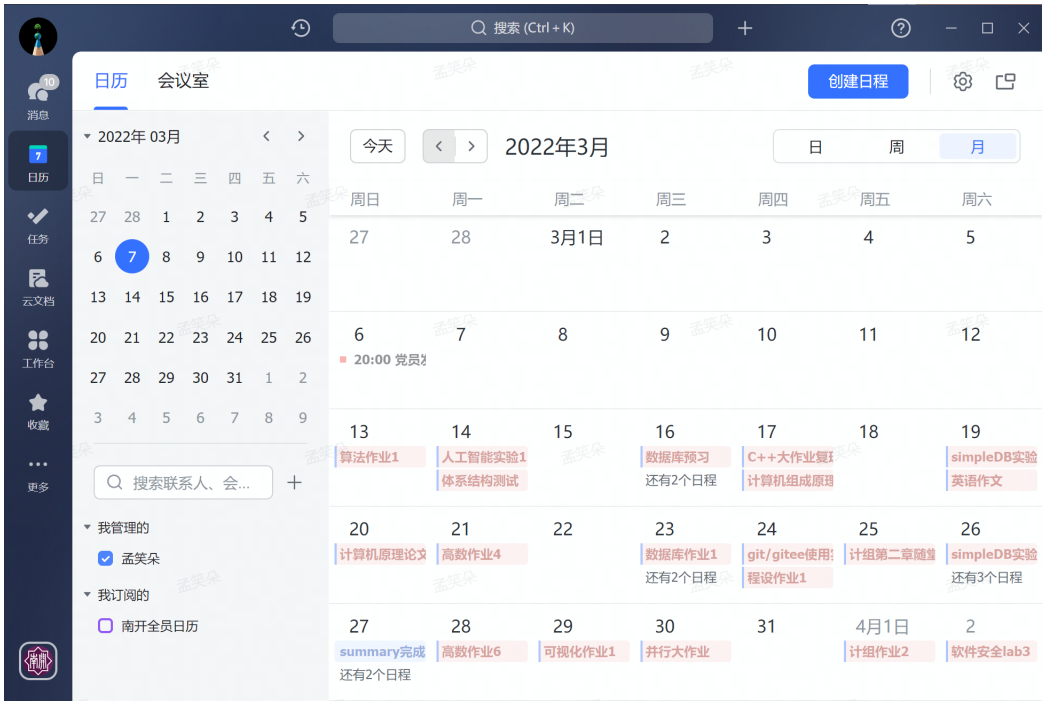
1 项目缘起

笔者是一个热衷于做计划的人，信奉“无计划不行动”，习惯于做事情之前写一个计划列表，从初一到高三都乐此不疲地在便利贴上，在笔记本上写下自己的学习计划与对未来的规划，上大学以来，我也探索了不同的计划任务软件，这些笔记软件有的以目标为导向，有的以过程为驱动，有的面面俱到，有的简洁到只是为了备忘。在这么多年坚持不懈的做计划过程中，我也摸索到了自己喜欢的计划风格，也逐渐明确了自己想要的是什么样的计划软件。

现在是一个人人彰显个性的时代，越来越强调人自身的独特性，这就使得产品的需求逐渐向个性化，定制化靠拢，传统的大众型的产品对于追求个性化的一部分人来说已很难满足要求。实现 MyAutoCalendarToDo 这个项目的初心就是为了个性化定制属于我的日程管理软件，并借此探索互联网产品软件的个性化设计。

2 项目的设计思路

笔者使用过各种任务管理软件之后发现，飞书日历这类日历型的软件对于我来说是最好用的，一方面可以起到提醒的作用，不至于忘记了某个任务的截止时间；另一方面，可以宏观地整体地来把控需要完成的任务，不至于陷入计划任务的细节当中，反而在做计划的时候浪费了大量的时间。



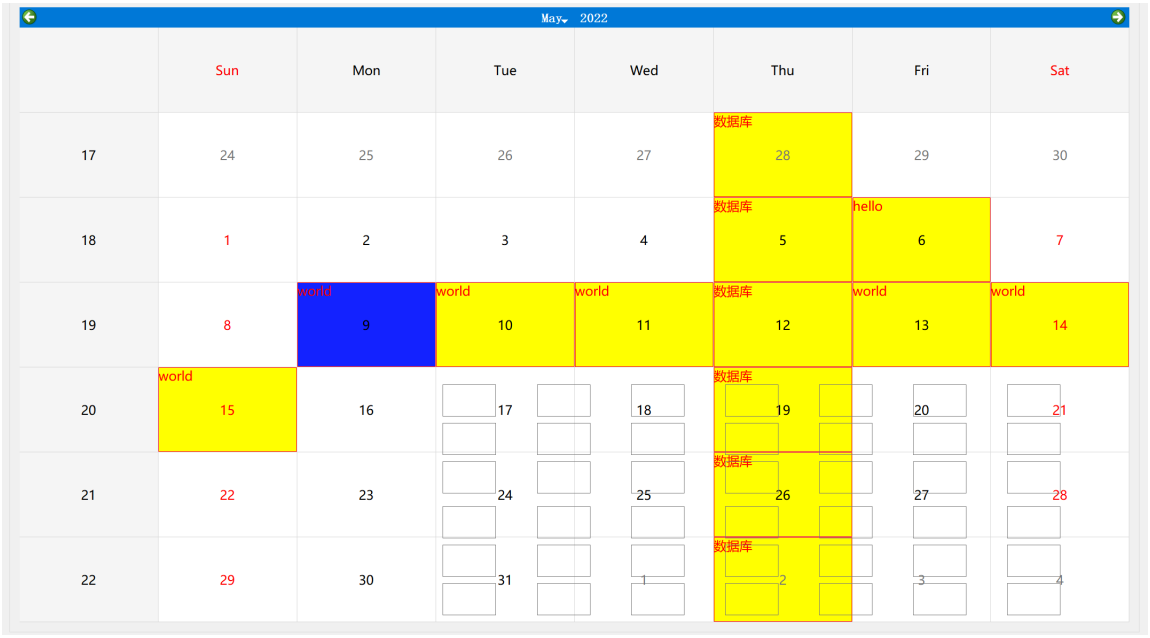
但是这种日历型的计划软件有一个缺点：常常会有这样一些时间节点，分布着大量任务的截止时间，在这些时间节点之前如果不能对任务进行合理的计划管理，邻近时期往往很容易造成任务完成的失衡或者手忙脚乱的情况，比如下面这种情况：

在这种情况下，设计高效的可替代人力的自动化任务规划算法，指定各项任务的截止时间，然后进行算法设计将会节省大量时间，省去我们进行任务规划所花费的时间。项目目标就明确了：

1. 实现一个日历类型的可视化任务管理界面；
2. 实现给定任务自动分配每日任务；

在设计一个日历类型的可视化任务管理界面时，笔者发现这并不容易，相比于飞书日历的界面和

实用性，自己写的日历可视化界面无论是在美观性，还是可靠性方面都不如飞书日历，当实现了第一项之后发现第二项的实现异常困难，下面这张图是笔者实现自定义的日历的截图：



注：以上界面实现借鉴了 https://www.writebug.com/git/Ridiculous/calendar-program/src/branch/master 源码

可以发现不仅界面不够美观，甚至程序会出现某种异常，使得显示异常；而且本身程序就不稳定，再将第二点的算法加进去，程序会更加不稳定。不得已笔者只好放弃了这种实现思路。

经过一番寻找后，发现了和飞书日历类似甚至更加强大的谷歌日历，谷歌日历提供了 api 接口，可以在不打开 google calendar 的情况下，对日历各种操作。

唯一不足的是 google calendar api 必须借助 VPN 进行使用，经过一番思考后，笔者还是决定在 google calendar 的基础上进行 MyAutoCalendarToDo 的开发设计，这时的项目目标就变为：

1. 使用 google calendar api 连接到 google calendar，并使用 api 接口对 google calendar 进行一系列操作；

2. 利用 google calendar api 借助算法实现自动分配每日任务；

3 项目设计过程

3.1 利用 google calendar api 连接到 google calendar

3.1.1 如何使用 google calendar api？

查阅 Google API 开发手册得知，要使用 google celendar api 必须首先开启 calendar api 服务，然后根据开启服务的访问凭据动态获取登录的 token 令牌，借助 token 令牌来使用 api 服务。细化来讲，分为以下几个部分：

1. 配置 Google Cloud 项目和应用程序：在开发期间，需要在 Google Cloud Console 中注册应用程序，定义授权范围和访问凭据，以使用 API 密钥、最终用户凭据或服务帐户凭据对应用程序进行身份验证。

2. 验证应用程序的访问权限：当应用程序运行时，将评估注册的访问凭据。如果应用程序正在以最终用户身份进行身份验证，则可能会显示登录提示。

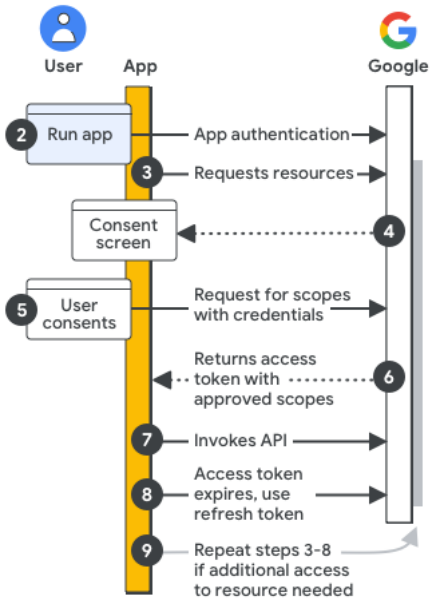
3. 请求资源：当应用需要访问 Google 资源时，它会使用之前注册的相关访问范围向 Google 询问。

4. 征求用户同意：如果应用以最终用户身份进行身份验证，Google 会显示 OAuth 同意屏幕，以便用户决定是否授予应用访问所请求数据的权限。

5. 发送已批准的资源请求：如果用户同意访问范围，应用会将凭据和用户批准的访问范围捆绑到请求中。该请求被发送到 Google 授权服务器以获取访问令牌。

6. Google 返回访问令牌：访问令牌包含授予的访问范围列表。如果返回的范围列表比请求的访问范围更受限制，应用程序将禁用任何受令牌限制的功能。

7. 访问请求的资源：应用程序使用来自 Google 的访问令牌来调用相关 API 并访问资源。

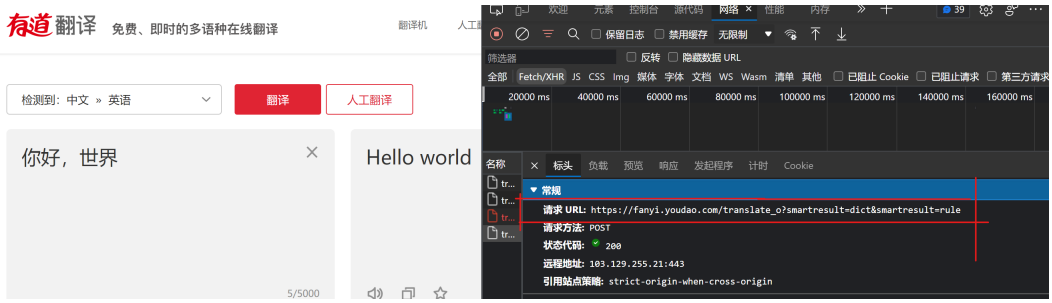


笔者已开通 google calendar api 的访问权限，关键是如何代码实现令牌的获取，实际上，我们有一个类是专门处理 token 令牌获取的，这个类就 QOAuth2AuthorizationCodeFlow 类，只需要将 client_scret 参数 load 到类中，我们就可以根据类内的一些方法来实现，下面是 GoogleQauth 类中的一些方法以及其功能：

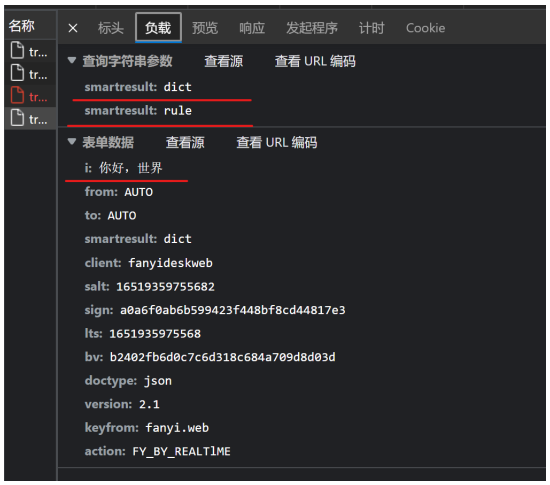
方法	功能
initialOAuth()	初始化
onGoogleGranted()	谷歌授权参数传递
onGoogleAuthorizeWithBrowser()	浏览器打开谷歌授权界面
accessToken()	获取令牌

3.1.2 使用 google calendar api 的一系列方法

这里要解释一下什么是 api，这里我们拿有道翻译来进行简单说明：



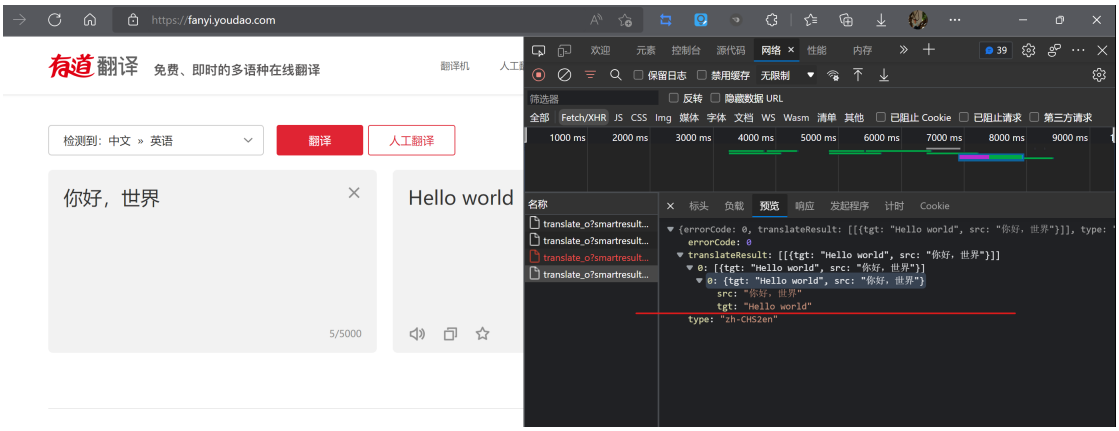
当我们进行翻译时，会有一个请求的 url，这里是 post 请求，我们的 calendar api 用的也是 post 请求，这种请求的话只需要记住要有参数传递进去，根据传递的参数返回不同的结果。当传递进去参数后：



再来看一下结果：

可以看到 src 就是我们传入的参数，tgt 就是结果的输出，这就启发我们只要知道 post 之后的 url 的信息中各种 key 值代表的含义就可以把对应的值取出来，api 的使用就是这个原理，下面结合本项目查看：

```
1  //获取 calendar
2  void CalendarDataManager::getCalendars(const QString& access_token)
3  {
4      qDebug() << Q_FUNC_INFO;
5      QString s = QString("https://www.googleapis.com/calendar/v3/users/me/calenda
```




```
    rList?access_token=%1").arg(access_token);//传入参数
    QApplication::setOverrideCursor(Qt::WaitCursor);
    m_pNetworkAccessManager->get(QNetworkRequest(QUrl(s)));//发送给网络端
}

//新日历
void CalendarDataManager::newCalendar(const QString& access_token, const QString & name)
{
    QString s = QString("https://www.googleapis.com/calendar/v3/calendars?access
    _token=%1").arg(access_token);

    QByteArray params;
    params.append(QString("{ \"summary\": \"%1\" }").arg(name).toUtf8());
    QNetworkRequest request;
    request.setUrl(QUrl(s));
    request.setRawHeader("Content-Type", "application/json");

    QApplication::setOverrideCursor(Qt::WaitCursor);
    m_pNetworkAccessManager->post(request, params);
}
```

还有一些其他的例子也是同样的道理，参考官方文档就可以学习到。另外还有一个很重要的方法就是要将上述发送给网络端中的数据提取出来：

```
void CalendarDataManager::replyFinished(QNetworkReply * reply)
{
    QApplication::restoreOverrideCursor();
    qDebug() << __FUNCTION__;
    QString json = reply->readAll();
    qDebug() << "Reply = " << json;
    qDebug() << "URL = " << reply->url();
```

```
        QString strUrl = reply->url().toString();
        ……
    }//这里的 url 就是后面有一大堆 key-value 值的 json 形式的内容，
    只要知道key 值就可以提取出具体的值。
```

这样，我们就完成了通过 api 和 google calendar 进行连接了，下面进行每日自动任务分配。

3.2 最小延迟调度算法及其扩展实现自动分配每日任务

首先，我们需要对功能细节化，针对我们的需求，对实际问题进行抽象：

设定每一件待办事项的截止时间，设定每一件待办事项的估计完成时间，然后自动进行每一天的任务划分。简单起见，我们先来实现紧凑模式，即每一天的空闲时间都被利用，没有空闲时间没有安排任务的情况。

设定条件：

1. 每天的空闲工作时间是 9h，每天的可进行的任务数量为 3；
2. 指定任务的截止时间与任务的持续时间；

在算法导论中，我们对单资源的最小化延迟调度问题做了探讨与证明，证明优先完成截止时间最近的项目，最后的延迟最小，这里不详细探讨了。

而对于这个问题来说，实际上我们有多个资源，即每次可以加三个任务，只需要每次将前三个截止时间最近的项目加入到结果中，最后遍历得到的一定是最优的解决方案。

当然，我们已经将问题进行最简化了，实际上，我们还有其他可以探讨的问题：

1. 如果考虑周六日和工作周的任务量不一样怎么办?
2. 如果考虑到上课工作等无空闲的时间怎么办?
3. 如果考虑到不同事项的优先级不一样怎么办?

预计笔者将在算法导论大作业中继续探讨这类问题，在此不做说明讨论。

基于以上思路，算法设计思路如下：

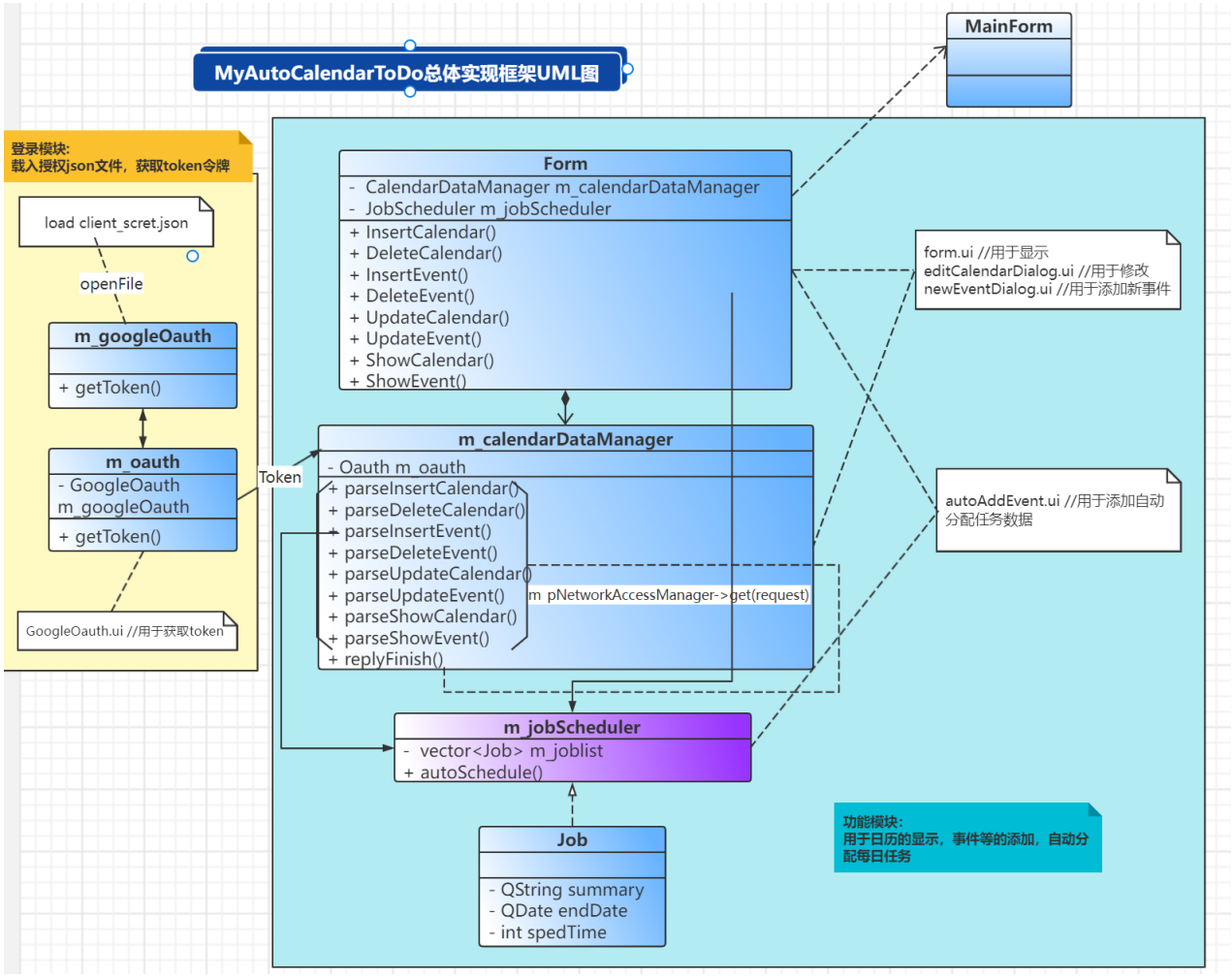
```
void JobScheduler::JobScheduler::AutoSchedule(){
    sort(joblist.begin(),joblist.end(),cmp);
    while (joblist.size() > 0)
    {
        resultList1.push_back(joblist[0].getSummary());
        joblist[0].changeNumber(-1);
        if (joblist.size() > 1) { resultList2.push_back(joblist[1].getSummary());
        joblist[1].changeNumber(-1); }
        if (joblist.size() > 2) { resultList3.push_back(joblist[2].getSummary());
        joblist[2].changeNumber(-1); }

        if (joblist.size() > 2)
        {
            if (joblist[2].getNumber() == 0)joblist.erase(joblist.begin() + 2);
        }
```

```
16            if (joblist.size() > 1)
17            {
18                if (joblist[1].getNumber() == 0)joblist.erase(joblist.begin() + 1);
19            }
20            if (joblist[0].getNumber() == 0)
21                joblist.erase(joblist.begin());
22        }
23    }
```

当然，其中还要包括与 UI 界面进行交互，进行事件的自动插入等工作，琐碎且繁杂，具体就不一一列举了。

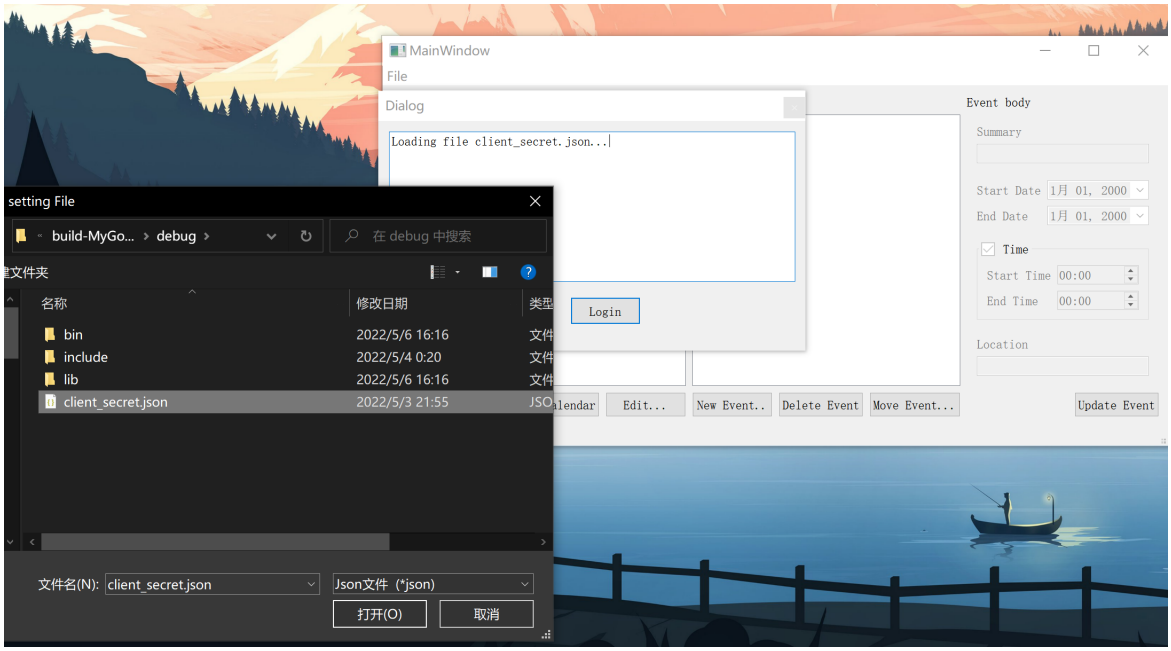
4 总体设计 UML 图

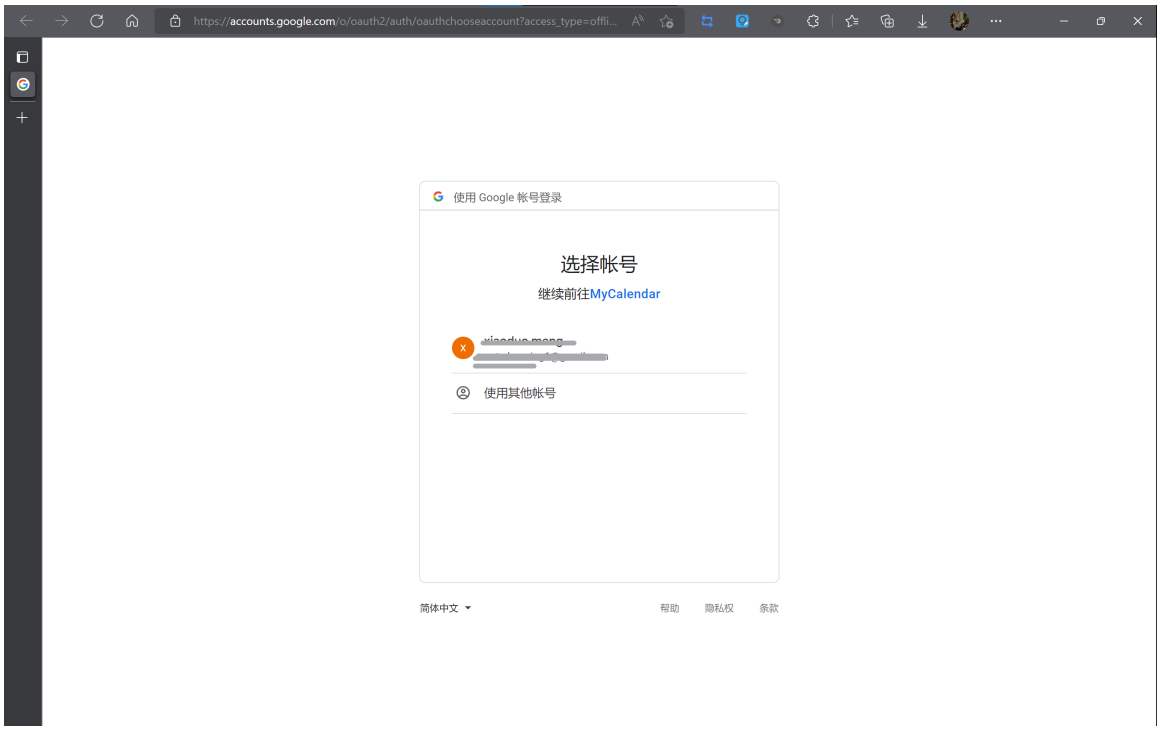
经过以上分析，可以将设计思路分为两个模块：第一个模块是登录模块，第二个模块是功能模块；各自的包含的类和功能如下图所示：



5 实现效果图

我们的程序截图：

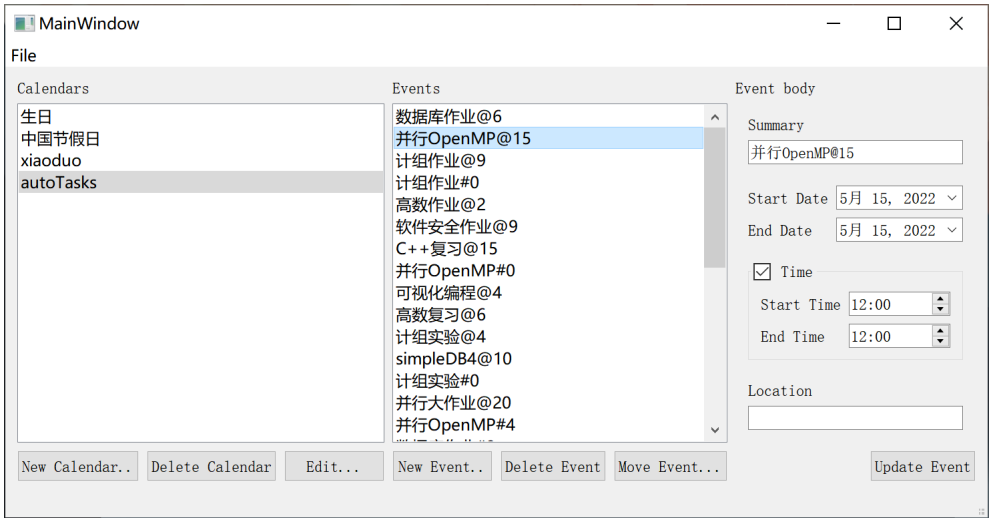


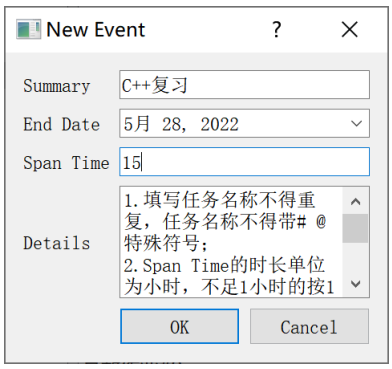


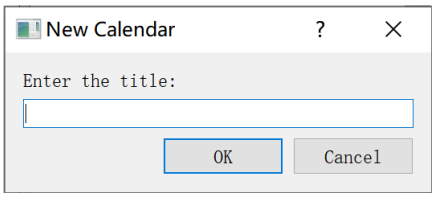
这是我们在 autoTasks 中利用自动分配任务算法生成的每日任务图

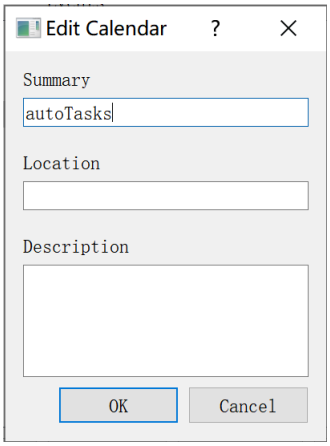
对比一下原来的飞书日历

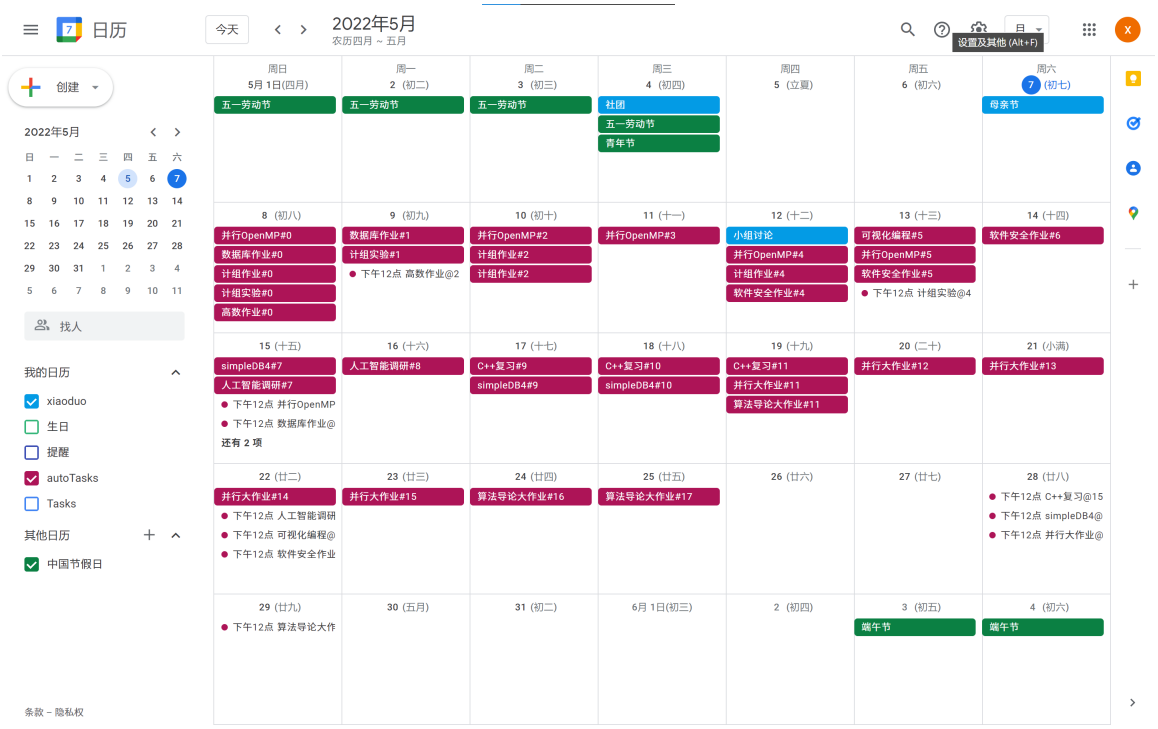
对比飞书日历，任务堆叠，很容易造成短时间内任务聚集，计划完成任全凭喜好和“感觉”，而自动生成任务之后，只需要专注于每一天要完成的任务就可以了，完成任务的效率更高。

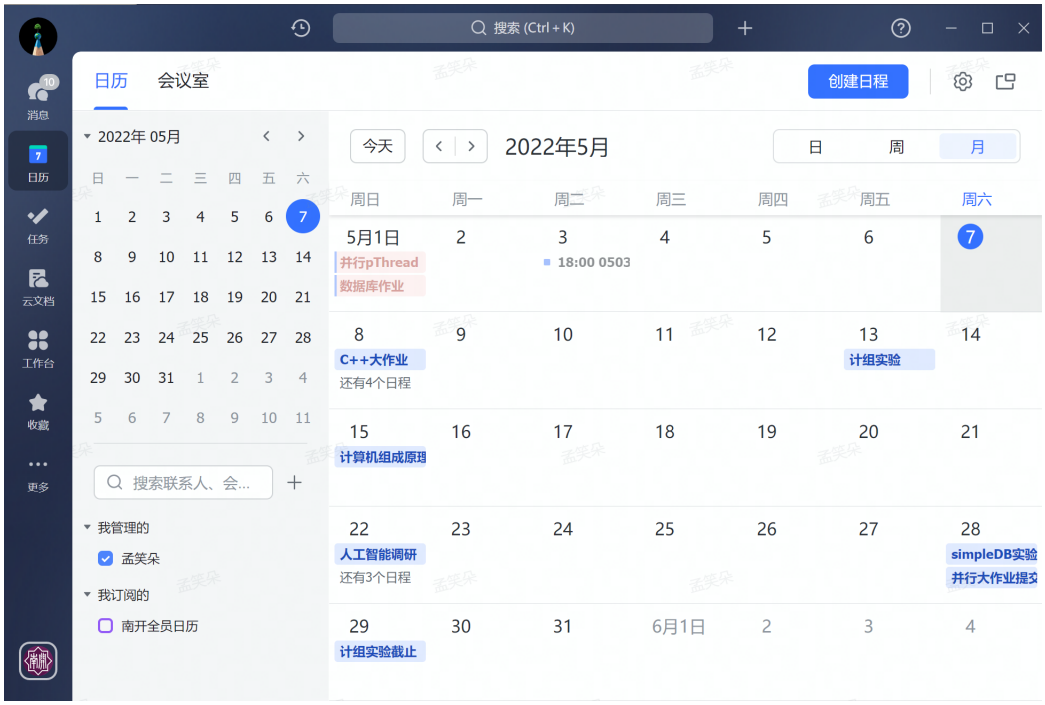

MainWindow
File
Calendars
生日
中国节假日
xiaoduo
autoTasks
Events
数据库作业@6
并行OpenMP@15
计组作业@9
计组作业#0
高数作业@2
软件安全作业@9
C++复习@15
并行OpenMP#0
可视化编程@4
高数复习@6
计组实验@4
simpleDB4@10
计组实验#0
并行大作业@20
并行OpenMP#4
Event body
Summary
并行OpenMP@15
Start Date 5月 15, 2022
End Date 5月 15, 2022
Time
Start Time 12:00
End Time 12:00
Location
New Calendar..
Delete Calendar
Edit...
New Event..
Delete Event
Move Event...
Update Event


New Event
Summary C++复习
End Date 5月 28, 2022
Span Time 15
Details
1.填写任务名称不得重复，任务名称不得带# @特殊符号;
2.Span Time的时长单位为小时，不足1小时的按1
OK
Cancel


New Calendar
Enter the title:
OK
Cancel


Edit Calendar
Summary
autoTasks
Location
Description
OK
Cancel


日历
今天
2022年5月
农历四月 ~ 五月
设置及其他 (Alt+F)
创建
2022年5月
找人
我的日历
xiaoduo
生日
提醒
autoTasks
Tasks
其他日历
中国节假日
条款 – 隐私权
周日 5月 1日(四月)
周一 2 (初二)
周二 3 (初三)
周三 4 (初四)
周四 5 (立夏)
周五 6 (初六)
周六 7 (初七)
五一劳动节
社团
青年节
母亲节
并行OpenMP#0
数据库作业#0
计组作业#0
计组实验#0
高数作业#0
数据库作业#1
计组实验#1
并行OpenMP#2
计组作业#2
并行OpenMP#3
小组讨论
并行OpenMP#4
计组作业#4
软件安全作业#4
可视化编程#5
并行OpenMP#5
软件安全作业#5
软件安全作业#6
simpleDB4#7
人工智能调研#7
人工智能调研#8
C++复习#9
simpleDB4#9
C++复习#10
simpleDB4#10
C++复习#11
并行大作业#11
算法导论大作业#11
并行大作业#12
并行大作业#13
并行大作业#14
并行大作业#15
算法导论大作业#16
算法导论大作业#17
端午节


搜索 (Ctrl + K)
日历
会议室
创建日程
消息
日历
任务
云文档
工作台
收藏
更多
2022年 05月
今天
2022年5月
日
周
月
周日
周一
周二
周三
周四
周五
周六
5月1日
并行pThread
数据库作业
18:00 0503
C++大作业
还有4个日程
计组实验
计算机组成原理
人工智能调研
还有3个日程
simpleDB实验
计组实验截止
搜索联系人、会...
我管理的
孟笑朵
我订阅的
南开全员日历

6 总结

在这次项目的实现过程中，经历了一系列的曲折，例如最初的 Oauth 令牌验证，笔者最初是用 QWebView 类来尝试实现的，结果查阅这篇文章得知《How To Authenticate with Google SSO in Qt (C++)》https://appfluence.com/productivity/how-to-authenticate-qt-google-sso/ ，在去年 Google 宣布不支持 QWebView 来获取 Oauth 验证，不得不另辟蹊径，结合这篇文章的内容和另一篇文章 https://www.qt.io/blog/2017/01/25/connecting-qt-application-google-services-using-oauth-2-0，实现了使用 QOAuth2Authorization 来实现授权。此外，Qt 中额外的包和环境的配置也是困扰我很长时间的一个问题，当初因为要使用到 QJson 这个类，反反复复卸载下载 Qt，从 Qt4 到 Qt5 至少尝试了十几次，切切实实体会到了用 10% 的时间来开发程序，用 90% 的时间来处理 bug。在算法的验证方面，因为 Qt 中编译时有编译器的多线程优化，导致程序异常添加数据等等，更是调试了不计其数次。

尽管如此，当程序成功地运行起来的时候，还是感觉到了一种巨大的满足感，在本项目中只是初步实现了 MyAutoCalendarToDo 的基本功能，距离成熟的产品还相距很远很远，功能只是基本满足了要求，界面也不够美观，程序的健壮性也欠缺考量，但笔者会一直将它完善下去，直至成为成熟的产品。

对于未来的学习，我也希望如此，当有想实现的小项目时，一定要动手去做，哪怕最初没有灵感，在不断地试错中一定会找到解决的思路的，同样，当完成之后，自己再有哪怕一点想法也要精益求精地做下去。

7 参考鸣谢

[1]https://www.writebug.com/git/Ridiculous/calendar-program/src/branch/master
[2]https://www.qt.io/blog/2017/01/25/connecting-qt-application-google-services-using-oauth-2-0
[3]https://developers.google.com/identity/protocols/oauth2/native-appcreatingcred
[4]https://appfluence.com/productivity/how-to-authenticate-qt-google-sso/
[5]https://cloud.google.com/docs/authentication/end-user
[6]https://github.com/chilarai/Qt-Google-OAuth
[7]https://github.com/tranter/qt-google-calendar
[8]《算法导论》