



实训 Project

[SmallGoal]

软件设计文档

队 长: 吴锐红

队 员: 孙宏川 候红潇 李明帆



目录

目录

一、项目分工.....	2
二、开发环境&架构设计.....	3
2.1. 开发环境.....	3
2.2. 架构描述.....	3
2.3. 架构图.....	4
三、项目阐述.....	4
3.1.项目简介.....	4
3.2.功能介绍.....	4
3.3.特色分析.....	6
四、项目展示.....	7
4.1.主页.....	7
4.2.Database 数据库.....	11
4.3.Media 媒体.....	15
4.4.File managerment 文件管理.....	17
4.5.Network accessing 网络通信.....	19
4.6.计划管理页面展示.....	20
4.7.Adaptive UI.....	25
4.8 目标编辑页面.....	26
4.9 具体目标时间详情页面.....	33
4.10 App to app communication.....	39
五、知识点的实现.....	41
5.1.Database 数据库.....	41
5.2.Media 媒体.....	42
5.3 File managerment 文件管理.....	44
5.4 Network accessing 网络通信.....	44
5.5 DataBinding 计划管理页面的数据绑定.....	46
5.6.Calendar 控件的使用.....	49
5.7 磁贴.....	49
5.8.磁贴循环展示.....	50
5.9.主页 DataBinding 部分.....	50
5.10 Adaptive UI 部分.....	51
5.11. App to app communication 部分.....	51
六、项目难点及解决方案.....	52



一、项目分工

学号	名字	角色	职责	贡献
15331315	吴锐红	组长	1. 负责作品功能和原型设计；负责可行性分析和市场分析 2. MainPage+live tiles; 3. MainPage的databinding 部分以及相关内容 4. MainPage (databinding、livetiles) 的项目展示、项目难点和解决方案 5. 软件设计文档&录制视频	26%
15331284	孙宏川	组员	1. timePageDetail+live tiles 2. timePageDetail的databinding 部分以及该相关内容 3. timePageDetail项目展示、项目难点和解决方案 4. 响应式 UI+app to app communication 分享目标 5. 安装部署说明&软件设计文档	24%
15331104	侯红潇	组员	1. goalEditPage实现; 2. goalEditPage的databinding 部分以及相关内容; 3. goalEditPage的项目展示、项目难点和解决方案 3. database、media 和 filemanagement 4. 页面跳转, Model, viewModel 5. 软件需求规格说明书&软件设计文档	26%
12330169	李明帆	组员	1. 负责作品实现特色和难点; 2. planPage (日计划页面+月计划页面+年计划页面); 3. timePage+live tiles 4. planPage和timePage的databinding 部分以及相关内容; 5. planPage和timePage的项目展示、项目难点和解决方案 6. 软件设计文档	24%

二、开发环境&架构设计

2.1. win10+visual studio 2015, c#+xaml 语言

2.2. 架构描述

该“SmallGoal”将使用MVC模式进行架构设计。MVC 模式代表 Model-View-Controller（模型-视图-控制器）模式，用于应用程序的分层开发，是一种软件设计典范。它是用一种业务逻辑、数据与界面显示分离的方法来组织代码，将众多的业务逻辑聚集到一个部件里面，在需要改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑，达到减少编码的时间。

1.Model（模型） - 模型代表一个存取数据的对象。模型对象负责封装及处理数据。通知视图改变。

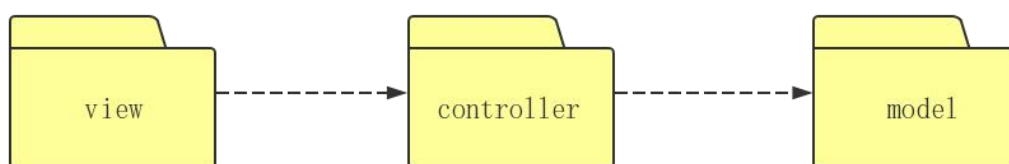
本例中，模型存储了相关计划和提醒的数据，如日计划月计划年计划等。。

2.View（视图） - 视图代表模型包含的数据的可视化。在本网站中由网页文件组成，负责获取用户行为并发送给控制器及显示控制器处理的结果。

本例中，视图层由首MainPage，TimePage，PlanePage，TimePageDetail和goalEditPage组成。

3.Controller（控制器） - 控制器作用于模型和视图上。它接收来自视图的请求并控制数据流向模型对象，更新模型对象状态，控制器选择用哪个视图显示模型更新的结果。它使视图与模型分离。

2.3. 架构图



三、项目阐述

3.1.项目简介

我们的软件是一个时间和目标管理软件。

目标管理，让用户能够有效地管理和清晰自己的目标；

时间管理——当任务截止日期到来时，给用户进行温馨的提醒以及对用户的目标进行时间统计，让用户对自己的目标更加重视以及清晰自己的时间分配；

计划管理，让用户能够宏观地看到自己的目标在日、周、月、年里面的分配，能够让用户为达

成自己的目标理想而制定有效的计划，因此我们的产品“小目标”就这样一步一步帮助用户管理自己的小目标，见证用户完成自己的一个一个小目标，最后实现自己的人生大目标。

3.2.功能介绍

本产品功能主要是三大管理功能：目标管理，时间管理，计划管理；

- (1) 目标管理是本产品的重要功能，分为以下子功能：
 - I. 创建目标，可以直接创建大目标，也可以在其他目标的基础上创建子目标；
 - II. 删除目标，用户可以删除目标；
 - III. 修改目标，用户修改目标信息；
 - IV. 完成目标，用户完成目标后可以点击完成，系统会记录下用户目标的完成情况；
- (2) 时间管理主要是附加在目标管理上面的，分为以下功能：
 - I. 每一个目标可以确定自己开始的时间以及想要完成的时间；
 - II. 可以通过计时功能对目标进行计时，当用户开始实现自己的目标时，通过我们的产品计时，我们会提供用户的已用时间、还需时间给用户，让其对自己的时间花费更清晰；
- (3) 计划管理：
 - I. 根据各个目标开始结束时间，可以生成该用户的日计划表，周计划表，月计划表，年计划表；

目标管理	<ol style="list-style-type: none">(1) 点击加号可以进入目标编辑界面，通过选择填写时间，备注之后可以创建目标。(2) 点击 checkbox 可以完成目标。(3) 点击右边的闹钟可以进入计时和相应目标的时间管理页面。(4) 点击目标进入目标编辑页面可以更新删除目标
时间管理	<ol style="list-style-type: none">(1) 一个列表呈现所有目标的预计花费时间以及已付出时间(需要通过计时)，还需要付出时间信息。(2) 点击目标 item 可以查看具体信息，例如完成进度，是否超时等一些客观事实同时可以计时。
计划管理	<ol style="list-style-type: none">(1) 计划以日历形式呈现。(2) 计划可分为月计划周计划年计划表，明晰易管理。

3.3.特色分析

3.4.1. 解决时间花费与安排问题——时间管理

我们经常问时间都去哪了？在生活中，我们常常很难计算出自己花费了多长的时间在某一件事情上。制定好的计划，一键可开启计时，在完成任务时点击结束即可自动将时间数据保存，方便易懂易操作。

3.4.2. 解决计划凌乱难以保存，难以坚持——计划管理

生活中常常面临着写好的纸质版计划无处可寻，计划写好却不能坚持实施，这款软件制定的计划清单功能以及计划时间结合功能可以很好地解决上述问题。

3.4.3. 方便的目标管理，一目了然——目标管理

设置一个个目标，目标以列表形式呈现，同时方便增删改查。提供目标时间管理和目标分日、月、年分类功能，让列表重要信息更好地呈现。

3.4.4. 迎合社会热点

在竞争激烈的社会，我们最缺的不是金钱，往往是时间。这款软件在一定程度上让用户更容易把握时间的节奏。

3.4.5. 可跨平台的 win10 应用。

Windows10 应用的可迁移性，扩大了这款应用的可使用范围与群体。



四、项目展示

4.1.主页

点击左上角的+进入目标编辑页面





在目标编辑界面添加新的计划，点击右下角的✓ 创建成功，返回主页。

SmallGoal

011 001

目标编辑

目标

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年 5月 12日

时间

14 00

结束日期

2017年 5月 12日

时间

14 00

备注

取消 ✓

SmallGoal

001 001

主页

+ 城市 提交

☐ todo1


☐ todo2

☐ todo3

☐ todo4

主页 时间管理 计划管理

主页提供了一个查询城市天气状况的接口，输入城市名称则可查询当日天气。



城市 多云 25°C

在主页点击任一计划的时钟按钮，进入具体目标时间详情界面。



SmallGoal 007 001 具体目标时间详情

todo1

预计花费时间：2小时0分钟

已付出时间： 天 : :

还需付出时间：2小时0分钟

起始时间：2017年5月12日 13:58

结束时间：2017年5月12日 15:58

▶

todo1

🔊 ▶ ...

点击信封的按钮，可进行共享。



SmallGoal 000 001 主页

+ 城市

- ☐ todo1
- ☐ todo2
- ☐ todo3

🏠 主页 🕒 时间管理 📅 计划管理

邮件

您的 Qq 帐户设置已过期。

格式 插入 选项 放弃 发送

B I U 标题 1 ↶ 撤回 ↷

发件人: 772604534@qq.com

收件人: 抄送和密件抄送

我的目标

目标: todo1

开始时间: 2017 年 5 月 12 日 13:58

结束时间: 2017 年 5 月 12 日 15:58

预计花费时间: 2 小时 0 分钟

已付出时间: 0 小时 0 分钟

还需付出时间: 2 小时 0 分钟

备注: todo1

发送自 Windows 10 邮件应用



点击页面下方时间管理按钮进入时间管理界面，该界面罗列了各个计划的进展程度。



点击任一计划进入具体目标时间详情界面。



4.2.Database 数据库:

使用了 SQLite 轻量级数据库:

SQLite Expert Personal 4.2 (x64)

Database: SmallGoalDB Table: GoalItem File: C:_36d0c427-c899-4ba0-b645-218eaf0dc495_awb217f4mgzj4\LocalState\SmallGoalDB.db SQLite library: sqlite3.dll 3.18.0 [FTS3 FTS4 FTS5 RTREE] Style: Iceberg Classico

rowid	name	type	startYear	startMonth	startDay	startHour	startMinute	endYear	endMonth	endDay	endHour	endMinute	note	isFinished	usedYear	usedMonth	usedDay	usedHour	usedMin	usedSecond
1	日计划	0	2017	5	9	10	0	2017	5	9	22	0	日计划的详情	0	0	0	0	9	12	0
2	月计划	1	2017	5	9	10	0	2017	5	10	22	0	月计划的详情	0	0	0	0	9	12	0
3	年计划	2	2016	5	9	10	0	2016	6	9	22	0	年计划的详情	0	0	0	0	9	12	0

可以看到最开始数据库中有三个 item:

rowid	name	type	startYear	startMonth	startDay	startHour	startMinute	endYear	endMonth	endDay	endHour	endMinute	note	isFinished	usedYear	usedMonth	usedDay	usedHour	usedMin	usedSecond
1	日计划	0	2017	5	9	10	0	2017	5	9	22	0	日计划的详情	0	0	0	0	9	12	0
2	月计划	1	2017	5	9	10	0	2017	5	10	22	0	月计划的详情	0	0	0	0	9	12	0
3	年计划	2	2016	5	9	10	0	2016	6	9	22	0	年计划的详情	0	0	0	0	9	12	0

在程序一启动, 程序便从数据库中加载数据, 可以看到程序中有三个 item:

SmallGoal

004 000

主页

十 城市 提交

☐ 日计划

☐ 月计划

☐ 年计划

主页 时间管理 计划管理



数据也是数据库中对应的数据：

SmallGoal

021 000

主页

目标编辑

+

城市

提交

☐ 日计划

☐ 月计划

☐ 年计划

目标

日计划

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年 5月 9日

时间

10 00

结束日期

2017年 5月 9日

时间

22 00

备注

日计划的详情

SmallGoal

011 000

主页

目标编辑

+

城市

提交

☐ 日计划

☒ 月计划

☐ 年计划

目标

月计划

☐ 日目标 ☒ 月目标 ☐ 年目标

开始日期

2017年 5月 9日

结束日期

2017年 5月 10日

备注

月计划的详情

SmallGoal

012 000

主页

目标编辑

+

城市

提交

☐ 日计划

☐ 月计划

☒ 年计划

目标

年计划

☐ 日目标 ☐ 月目标 ☒ 年目标

开始日期

2016年 5月

结束日期

2016年 6月

备注

年计划的详情



对日计划和月计划进行修改成如下：

SmallGoal 019 000

主页 目标编辑

+ 城市 提交

☐ 现操期中

☐ 月计划

☐ 年计划

目标

现操期中

☐ 日目标 ☒ 月目标 ☐ 年目标

开始日期

2017年 5月 12日

结束日期

2017年 5月 13日

备注

完成现操期中作业

删除 保存

SmallGoal 022 000

主页 目标编辑

+ 城市 提交

☐ 现操期中

☐ 现操实验报告

☐ 年计划

目标

现操实验报告

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年 5月 12日

时间

14 17

结束日期

2017年 5月 12日

时间

15 17

备注

完成现操实验报告

删除 保存



关闭程序，数据写入数据库中，可以看到，所有的改动都写入到了数据库中，在数据库中都得到了体现：

rowid	name	type	startYear	startMonth	startDay	startHour	startMinute	endYear	endMonth	endDay	endHour	endMinute
(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)
1	现操期中	1	2017	5	12	14	16	2017	5	13	14	1
2	现操实验报告	0	2017	5	12	14	17	2017	5	12	15	1
3	年计划	2	2016	5	9	10	0	2016	6	9	22	

下两张图，是对 item 的更清晰的截图，同时第二张是接在第一张后面的一些数据：

rowid	name	type	startYear	startMonth	startDay	startHour	startMinute	endYear	endMonth	endDay	endHour
(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)
1	现操期中	1	2017	5	12	14	16	2017	5	13	14
2	现操实验报告	0	2017	5	12	14	17	2017	5	12	15
3	年计划	2	2016	5	9	10	0	2016	6	9	22

endMinute	note	isFinished	usedYear	usedMonth	usedDay	usedHour	usedMinute	usedSecond
(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)	(empty)
16	完成现操期中作业	0	0	0	0	9	12	0
17	完成现操实验报告	0	0	0	0	9	12	0
0	年计划的详情	0	0	0	0	9	12	0

4.3.Media 媒体:

可以看到，在具体目标时间详情页面中，有一个媒体播放音乐，这一个功能是为了让用户能够一边计时工作，一边听音乐：



这一个媒体有音量、播放、暂停、停止和选取文件功能：





可以看到点击音量后，音量条出现，可以调节音量：

SmallGoal

005 001

具体目标时间详情

愉快地写作业

预计花费时间：8小时39分钟

已付出时间：

0

天

0

:

0

:

15

还需付出时间：8小时38分钟

起始时间：2017年5月12日 14:21

结束时间：2017年5月12日 23:00

☐

愉快地写着作业并且计时并且听歌

68

🔊

▶

⏸

□

📄

⋮

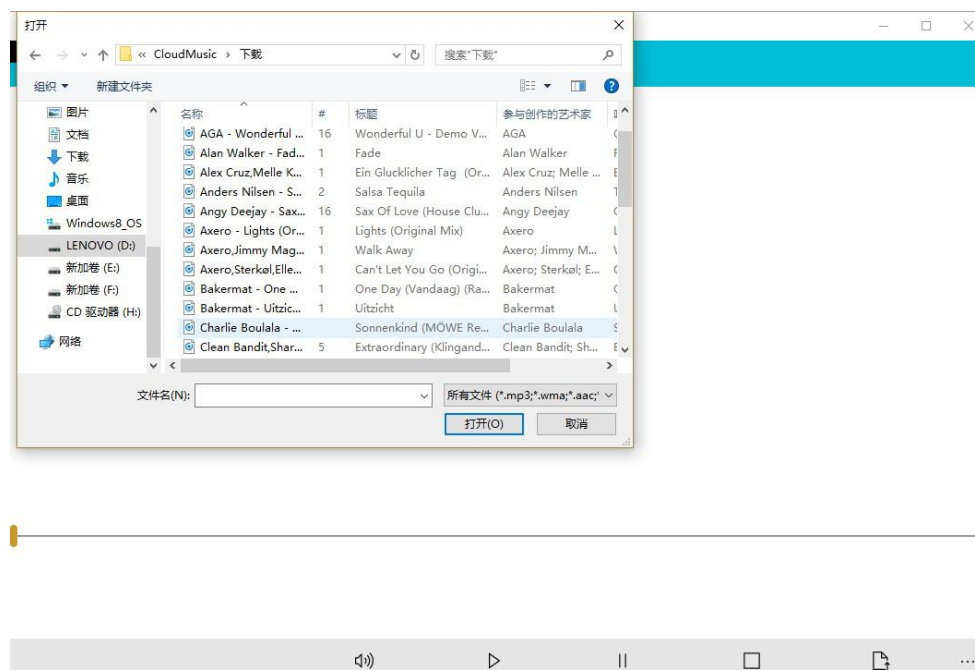


可以暂停、停止、播放：



4.4.File managerment 文件管理：

简单的文件管理功能嵌入在媒体中，即可以从外部选择媒体音频文件进行播放，点击媒体的选择文件按钮，可以出现选择界面：





选择一首歌，就可以在程序中进行播放了：

SmallGoal

— □ ×

011 001

具体目标时间详情

愉快地写作业

预计花费时间：8小时39分钟

已付出时间：

0

天

0

:

2

:

43

还需付出时间：8小时36分钟

起始时间：2017年5月12日 14:21

结束时间：2017年5月12日 23:00

☐

愉快地写着作业并且计时并且听歌

🔊 ▶ || □ 📄 ...

4.5.Network accessing 网络通信:

考虑到用户可能在每天记录自己的一天目标时，可能会看一看今天的天气，然后决定做一些事情，因此首页插入了天气查询功能，如图，输入要查询天气的城市，点击提交，会出现该城市的天气：



于是用户看到今天天气偏凉，就增加了出门带一件薄外套的 item:





4.6.计划管理页面展示:

计划管理页面首页:

SmallGoal

015 001

计划管理

年计划

月计划

日计划

	时间	计划完成事项	完成时长																																																	
<div>2017年5月</div> <table><thead><tr><th>日</th><th>一</th><th>二</th><th>三</th><th>四</th><th>五</th><th>六</th></tr></thead><tbody><tr><td>30</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td></tr><tr><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr><tr><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td></tr><tr><td>28</td><td>29</td><td>30</td><td>31</td><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr></tbody></table>				日	一	二	三	四	五	六	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10
日	一	二	三	四	五	六																																														
30	1	2	3	4	5	6																																														
7	8	9	10	11	12	13																																														
14	15	16	17	18	19	20																																														
21	22	23	24	25	26	27																																														
28	29	30	31	1	2	3																																														
4	5	6	7	8	9	10																																														

主页

时间管理

计划管理

点击一个日期, 会显示日计划:

SmallGoal

004 001

计划管理

年计划

月计划

日计划

	时间	计划完成事项	完成时长																																																	
<div>2017年5月</div> <table><thead><tr><th>日</th><th>一</th><th>二</th><th>三</th><th>四</th><th>五</th><th>六</th></tr></thead><tbody><tr><td>30</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td></tr><tr><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr><tr><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td></tr><tr><td>28</td><td>29</td><td>30</td><td>31</td><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr></tbody></table>				日	一	二	三	四	五	六	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10
日	一	二	三	四	五	六																																														
30	1	2	3	4	5	6																																														
7	8	9	10	11	12	13																																														
14	15	16	17	18	19	20																																														
21	22	23	24	25	26	27																																														
28	29	30	31	1	2	3																																														
4	5	6	7	8	9	10																																														

14时17分 - 15时17分

现操实验报告

0小时12分钟

14时21分 - 23时0分

愉快地写作业

0小时2分钟

14时28分 - 14时28分

出门带一件薄外套

0小时0分钟

主页

时间管理

计划管理



点击月计划，会显示该月的计划列表：

SmallGoal

000 000

计划管理

年计划

月计划

日计划

2017年5月

日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

待月计划列表

现操期中

月目标其一：将3d游戏课程完成

已完成月计划

主页

时间管理

计划管理



点击年计划会出现该年的计划列表：

SmallGoal

019 000

计划管理

年计划
月计划
日计划

2017年5月

日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

待年计划列表

年计划1：我要减肥

年计划2：我要努力编程

年计划3：我要达到跑半马的目标

已完成年计划

主页

时间管理

计划管理

增加了几个 2016 年的月计划和年计划，并且相应有完成和未完成进行测试：

- ☒ 2016年计划1：大二上取的好成绩 🕒 ✉
- ☐ 2016年计划2：去内蒙古旅游 🕒 ✉
- ☒ 2016月计划1：看夏洛克 🕒 ✉
- ☐ 2016月计划2：看黑镜 🕒 ✉



通过点击日历上方的标题进行选择其他的月：

2017年5月							^	v
日	一	二	三	四	五	六		
30	1	2	3	4	5	6		
7	8	9	10	11	12	13		
14	15	16	17	18	19	20		
21	22	23	24	25	26	27		
28	29	30	31	1	2	3		
4	5	6	7	8	9	10		

再通过左上方的点击选择其他的年份：

2017年				^	v
1月	2月	3月	4月		
5月	6月	7月	8月		
9月	10月	11月	12月		
1月	2月	3月	4月		

2010 - 2019				^	v
2009	2010	2011	2012		
2013	2014	2015	2016		
2017	2018	2019	2020		
2021	2022	2023	2024		



于是选择进入了 2016 年5 月12 日，点击了其月计划，可以看到：

SmallGoal

008 000

计划管理

年计划

月计划

日计划

待月计划列表

2016月计划2：看黑镜

2016年5月

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

已完成月计划

2016月计划1：看夏洛克

主页

时间管理

计划管理

点击年计划，可以看到：

SmallGoal

015 000

计划管理

年计划

月计划

日计划

待年计划列表

2016年计划2：去内蒙古旅游

2016年5月

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

已完成年计划

2016年计划1：大二上取的好成绩

主页

时间管理

计划管理



4.7.Adaptive UI

App 主界面宽度较小时，只显示主页；宽度较大时，还会同时显示目标编辑页面：



4.8 目标编辑页面

点击主页左上角的加号按钮，可跳转至目标编辑页面：

主页

+

城市 广州

提交

阵雨

25°C

☐ todo

🕒

✉

☐ todo

🕒

✉

目标编辑

目标

☒ 日目标

☐ 月目标

☐ 年目标

开始日期

2017年

5月

12日

时间

21

55

结束日期

2017年

5月

12日

时间

21

55

备注

✕ 取消

✓ 保存



目标编辑页面可创建新目标。其中，目标分为三种：日目标、月目标和年目标：

目标编辑

目标

新目标

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年

5月

12日

时间

21

46

结束日期

2017年

5月

12日

时间

23

45

备注

吃晚饭

取消

保存

目标编辑

目标

新目标

☐ 日目标 ☒ 月目标 ☐ 年目标

开始日期

2017年

5月

12日

结束日期

2017年

5月

13日

备注

吃晚饭

取消

保存



点击界面右下角保存按钮，创建该新目标



创建新目标过程中，若想退出，按界面右下角的取消按钮或左上角的返回按钮，可返回主页：

在主页点击一项目标，可进入目标编辑页面编辑该目标：



←

目标编辑

目标

新目标

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年

5月

12日

时间

21

56

结束日期

2017年

5月

12日

时间

23

55

备注

吃晚饭

删除

保存

对目标进行更改并保存：

←

目标编辑

目标

新目标----我要吃好吃的 ×

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年

5月

12日

时间

21

56

结束日期

2017年

5月

12日

时间

23

55

备注

吃晚饭

删除

保存

主页

+

城市

提交

☐

todo

☐

todo

☐

todo

☐

新目标---我要吃好吃的

主页

计划管理

若目标名字过长、备注过长，或者目标的结束时间早于开始时间，或者目标设置的时间不符合目标类型（日目标只能设置在同一天，月目标只能设置在同一个月，年目标只能设置在同一年），目标是不会被保存的：

提示

日目标只能设置在同一天哦(๖๖๖)...

请改为月（年）目标吧(^_^)

确定

提示

月目标只能设置在同一个月哦(๖๖๖)...

请改为年目标吧(^_^)

确定

提示

年目标只能设置在同一年哦(๖๖๖)...

请创建多个年目标吧(^_^)

确定

提示

结束早过开始啦(๖๖๖)...

确定



删除该目标，主页和时间管理页面将不再显示该目标：

←

目标编辑

目标

新目标----我要吃好吃的

☒ 日目标 ☐ 月目标 ☐ 年目标

开始日期

2017年

5月

12日

时间

21

56

结束日期

2017年

5月

12日

时间

23

55

备注

吃晚饭

删除

保存



4.9 具体目标时间详情页面

在时间管理页面点击一项目标或在主页点击该目标的计时按钮，可进入具体目标时间详情页面：

时间管理

新目标----吃晚饭 预计花费时间:1小时0分钟 已付出时间:0小时0分钟 还需要付出:1小时0分钟

todo 预计花费时间: 12小时0分钟 已付出时间: 9小时12分钟 还需要付出: 2小时48分钟

todo 预计花费时间: 365天12小时 已付出时间: 0天9小时 还需要付出: 365天2小时

todo 预计花费时间: 12小时0分钟 已付出时间: 9小时12分钟 还需要付出: 2小时48分钟

todo 预计花费时间: 31天12小时 已付出时间: 0天9小时 还需要付出: 31天2小时

todo 预计花费时间: 365天12小时 已付出时间: 0天9小时 还需要付出: 365天2小时

⌚

时间管理



主页

+

城市

提交

☐ 新目标----吃晚饭

☐ todo

☐ todo

☐ todo

☐ todo

🏠

🕒 时间管理

📅 计划管理

具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：

0

天

0

:

0

:

0

还需付出时间：1小时0分钟

起始时间：2017年5月11日 19:00

结束时间：2017年5月11日 20:00

▶

我要吃牛肉！

🔊 ...

具体目标时间详情页面显示的是该目标的详情：预计花费时间、已付出时间、还需付出时间等，并能计时（计算该目标以花费的时间）
点击开始按钮，开始计时：

←
具体目标时间详情

新目标----吃晚饭

预计花费时间： 1小时0分钟

已付出时间： 天 : :

还需付出时间： 1小时0分钟

起始时间： 2017年5月11日 19:00

结束时间： 2017年5月11日 20:00

▶

我要吃牛肉！

✕

SmallGoal

SmallGoal

SmallGoal

...

计时过程中，已付出时间一栏会动态地增加秒数，还需付出时间一栏也会动态地进行更新：

←
具体目标时间详情

新目标----吃晚饭

预计花费时间： 1小时0分钟

已付出时间： 天 : :

还需付出时间： 0小时59分钟

起始时间： 2017年5月11日 19:00

结束时间： 2017年5月11日 20:00

□

我要吃牛肉！

🔊

...

若用户为该目标付出了时间但忘记计时，或忘记停止计时导致已付出时间不必要地增多，可以点击已付出时间一栏中的 TextBox，对时间进行修改：

← 具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：0 天 0 : 30 : 20

还需付出时间：0小时29分钟

起始时间：2017年5月11日 19:00

结束时间：2017年5月11日 20:00

☐

我要吃牛肉！

🔊 ...

（说明：鼠标点击页面空白地方，使 TextBox 失去聚焦，时间就已经修改成功）

若输入非法字符，是会被接受的：

SmallGoal

← 具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：0 天 -1 : 31 : 38

还需付出时间：0小时28分钟

起始时间：2017年5月11日 19:00

结束时间：2017年5月11日 20:00

☐

我要吃牛肉！

🔊 ...



←

具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：天::

提示
非法输入！

确定

🔊

...

若用户启用了计时功能，中途离开该界面，计时还是会照常进行的：

←

具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：天::

还需付出时间：0小时25分钟

起始时间：2017年5月11日 19:00

结束时间：2017年5月11日 20:00

☐

我要吃牛肉！

🔊

...



时间管理

目标----吃晚饭 预计花费时间:1小时0分钟 已付出时间:0小时35分钟 还需要付出:0小时24分钟

todo 预计花费时间: 12小时0分钟 已付出时间: 9小时12分钟 还需要付出: 2小时48分钟

todo 预计花费时间: 365天12小时 已付出时间: 0天9小时 还需要付出: 365天2小时

todo 预计花费时间: 12小时0分钟 已付出时间: 9小时12分钟 还需要付出: 2小时48分钟

todo 预计花费时间: 31天12小时 已付出时间: 0天9小时 还需要付出: 31天2小时

todo 预计花费时间: 365天12小时 已付出时间: 0天9小时 还需要付出: 365天2小时



计划管理



具体目标时间详情

新目标----吃晚饭

预计花费时间: 1小时0分钟

已付出时间: 0 天 0 : 35 : 58

还需付出时间: 0小时24分钟

起始时间: 2017年5月11日 19:00

结束时间: 2017年5月11日 20:00



我要吃牛肉!





点击了停止按钮、或关闭该应用，计时才会停止：

← 具体目标时间详情

新目标----吃晚饭

预计花费时间：1小时0分钟

已付出时间：0 天 0 : 36 : 34

还需付出时间：0小时24分钟

起始时间：2017年5月11日 19:00

结束时间：2017年5月11日 20:00

☐

我要吃牛肉！

🔊 ...

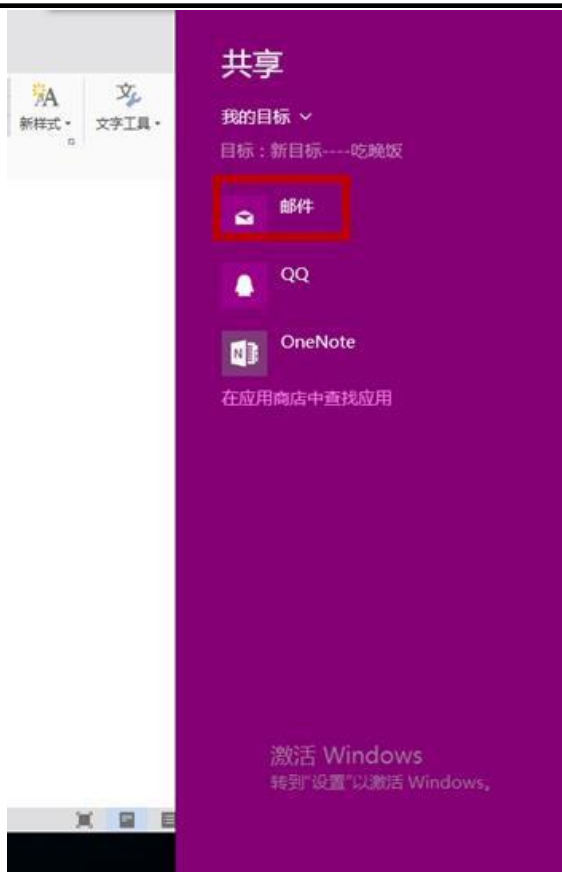
该应用会保存好已付出的时间：

新目标----吃晚饭 预计花费时间:1小时0分钟 已付出时间:0小时38分钟 还需要付出:0小时21分钟

4.10 App to app communication

在主页面点击一项目标的分享按钮，可将该目标的内容分享出去：

☐ 新目标----吃晚饭 ⌚ ☐



五、知识点的实现

5.1.Database 数据库

在App.xaml.cs 中，程序初始化时连接数据库：

```
public static SQLiteConnection database;
/// <summary>
/// 初始化单一实例应用程序对象。这是执行的创作代码的第一行，
/// 已执行，逻辑上等同于 main() 或 WinMain()。
/// </summary>
public App()
{
    this.InitializeComponent();
    this.Suspending += OnSuspending;
    database = new SQLiteConnection("SmallGoalDB.db");
}
```

在程序启动时加载数据库：

```
/*-----数据库部分-----*/
// 加载数据
private void LoadDatabase()
{
    // 如果表不存在则创建表，加载数据
    string sql = @"CREATE TABLE IF NOT EXISTS GoalItem (name VARCHAR( 20 ),
                                                         type INTEGER,
                                                         startYear INTEGER,
                                                         startMonth INTEGER,
                                                         startDay INTEGER,
                                                         startHour INTEGER,
                                                         startMinute INTEGER,
                                                         endYear INTEGER,
                                                         endMonth INTEGER,
                                                         endDay INTEGER,
                                                         endHour INTEGER,
                                                         endMinute INTEGER,
                                                         note VARCHAR( 200 ),
                                                         isFinished INTEGER,
                                                         usedYear INTEGER,
                                                         usedMonth INTEGER,
                                                         usedDay INTEGER,
                                                         usedHour INTEGER,
                                                         usedMinute INTEGER,
                                                         usedSecond INTEGER
                                                         );";

    using (var statement = App.database.Prepare(sql))
    {
        statement.Step();
    }
    // 将数据从数据库加载到程序中来
    using (var statement = database.Prepare("SELECT * FROM GoalItem"))
    {
        while (SQLiteResult.ROW == statement.Step())
        {
            myViewModel.AddMyGoalItem((string)statement[0], Convert.ToInt32((long)statement[1]), Convert.ToInt32((long)statement[2]), Convert.ToInt32((long)statement[3]),
                                       Convert.ToInt32((long)statement[4]), Convert.ToInt32((long)statement[5]), Convert.ToInt32((long)statement[6]), Convert.ToInt32((long)statement[7]), Convert.ToInt32((long)statement[8]),
                                       Convert.ToInt32((long)statement[9]), Convert.ToInt32((long)statement[10]), Convert.ToInt32((long)statement[11]), (string)statement[12], Convert.ToInt32((long)statement[13]),
                                       Convert.ToInt32((long)statement[14]), Convert.ToInt32((long)statement[15]), Convert.ToInt32((long)statement[16]), Convert.ToInt32((long)statement[17]), Convert.ToInt32((long)statement[18]));
        }
    }
}
```

在程序挂起时，保存数据到数据库中：

```
// 保存数据
private void SaveDatabase()
{
    deleteDataFromDataBase();
    for (int i = 0; i < myViewModel.allItems.Count; i++)
    {
        insertDataToDataBase(i);
    }
}

// 插入数据到数据库中
private void insertDataToDataBase(int i)
{
    using (var newGoalItem = database.Prepare("INSERT INTO GoalItem (name, type, startYear, startMonth, startDay, startHour, startMinute, endYear, endMonth, endDay, endHour, endMinute, note, isFinished, usedYear, usedMonth, usedDay, usedHour, usedMinute, usedSecond)"))
    {
        newGoalItem.Bind(1, myViewModel.allItems[i].name);
        newGoalItem.Bind(2, myViewModel.allItems[i].type);
        newGoalItem.Bind(3, myViewModel.allItems[i].startYear);
        newGoalItem.Bind(4, myViewModel.allItems[i].startMonth);
        newGoalItem.Bind(5, myViewModel.allItems[i].startDay);
        newGoalItem.Bind(6, myViewModel.allItems[i].startHour);
        newGoalItem.Bind(7, myViewModel.allItems[i].startMinute);
        newGoalItem.Bind(8, myViewModel.allItems[i].endYear);
        newGoalItem.Bind(9, myViewModel.allItems[i].endMonth);
        newGoalItem.Bind(10, myViewModel.allItems[i].endDay);
        newGoalItem.Bind(11, myViewModel.allItems[i].endHour);
        newGoalItem.Bind(12, myViewModel.allItems[i].endMinute);
        newGoalItem.Bind(13, myViewModel.allItems[i].note);
        newGoalItem.Bind(14, myViewModel.allItems[i].isFinished);
        newGoalItem.Bind(15, myViewModel.allItems[i].usedYear);
        newGoalItem.Bind(16, myViewModel.allItems[i].usedMonth);
        newGoalItem.Bind(17, myViewModel.allItems[i].usedDay);
        newGoalItem.Bind(18, myViewModel.allItems[i].usedHour);
        newGoalItem.Bind(19, myViewModel.allItems[i].usedMinute);
        newGoalItem.Bind(20, myViewModel.allItems[i].usedSecond);
        newGoalItem.Step();
    }
}

// 删除所有item
private void deleteDataFromDataBase()
{
    using (var statement = App.database.Prepare("DELETE FROM GoalItem"))
    {
        statement.Step();
    }
}
```

5.2.Media 媒体

在timePageDetail 页面中，即目标时间详情页面中，加入媒体和相应的控制按钮和转换器：

```
<Page.Resources>
    <local:DoubleToTimeSpan x:Key="DoubleToTimeSpan"/>
</Page.Resources>
```

```
<Page.BottomAppBar>
    <CommandBar>
        <CommandBar.Content>
            <Slider Grid.Row="2" Height="50" Name="volumeSlider" HorizontalAlignment="Left" VerticalAlignment="Center" ValueChanged="volumeSlider_ValueChanged"
                Minimum="0" Maximum="100" Value="50" Width="300" Margin="0,0,0,0" Opacity="0"/>
            </CommandBar.Content>
            <AppBarToggleButton Width="140" Height="30" Icon="Volume" Label="Volume" Click="VoiceButton_Click"/>
            <AppBarButton Width="140" Height="30" Icon="Play" Label="Play" Click="PlayButton_Click"/>
            <AppBarButton Width="140" Height="30" Icon="Pause" Label="Pause" Click="PauseButton_Click"/>
            <AppBarButton Width="140" Height="30" Icon="Stop" Label="Stop" Click="StopButton_Click"/>
            <AppBarButton Width="140" Height="30" Icon="OpenFile" Label="OpenFile" Click="pickFileButton_Click"/>
        </CommandBar>
    </Page.BottomAppBar>
```

```
<Grid Grid.Row="2">
    <MediaElement Grid.Row="0" Name="mediaPlayerElement" Source="Assets/Kindness.mp3" Margin="0,44,0,0" Position="{Binding Value, Converter={StaticResource DoubleToTimeSpan}, ElementName=timelineSlider}" />
    <Slider Grid.Row="1" Height="50" Name="timelineSlider" Margin="0,0,0,80"/>
</Grid>
```

这是上张图的后面部分：

```
ElementName=timelineSlider, Mode=TwoWay}" MediaOpened="mediaPlayerElement_MediaOpened"/>
```

在后台写控制媒体播放、暂停、停止、音量和选择文件的各个函数：

```
/*----- 媒体和文件管理部分 -----*/
bool isVolumeSlideShow = false;
private async void pickFileButton_Click(object sender, RoutedEventArgs e)
{
    // Create and open the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.ViewMode = PickerViewMode.Thumbnail;
    openPicker.SuggestedStartLocation = PickerLocationId.VideosLibrary;
    openPicker.FileTypeFilter.Add(".mp3");
    openPicker.FileTypeFilter.Add(".wma");
    openPicker.FileTypeFilter.Add(".aac");
    openPicker.FileTypeFilter.Add(".m4a");

    StorageFile file = await openPicker.PickSingleFileAsync();
    if (file != null)
    {
        var stream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read);

        mediaPlayerElement.SetSource(stream, file.ContentType); //MediaSource.CreateFromStorageFile(file);
        mediaPlayerElement.Play();
    }
}

private void PauseButton_Click(object sender, RoutedEventArgs e)
{
    mediaPlayerElement.Pause();
}

private void PlayButton_Click(object sender, RoutedEventArgs e)
{
    mediaPlayerElement.Play();
}

private void StopButton_Click(object sender, RoutedEventArgs e)
{
    mediaPlayerElement.Stop();
}

private void VoiceButton_Click(object sender, RoutedEventArgs e)
{
    if (isVolumeSlideShow == false)
    {
        volumeSlider.Opacity = 1;
        isVolumeSlideShow = true;
    }
    else
    {
        volumeSlider.Opacity = 0;
        isVolumeSlideShow = false;
    }
}

private void volumeSlider_ValueChanged(object sender, RangeBaseValueChangedEventArgs e)
{
    if (mediaPlayerElement != null) mediaPlayerElement.Volume = (double)(volumeSlider.Value / 100);
}

private void mediaPlayerElement_MediaOpened(object sender, RoutedEventArgs e)
{
    timelineSlider.Value = 0;
    timelineSlider.Maximum = mediaPlayerElement.NaturalDuration.TimeSpan.TotalSeconds;
}
}
```

转换器函数:

```
/*----- 转换器 -----*/  
class DoubleToTimeSpan : IValueConverter  
{  
    public object Convert(object value, Type targetType, object parameter, string language)  
    {  
        return TimeSpan.FromSeconds((double)value);  
    }  
  
    public object ConvertBack(object value, Type targetType, object parameter, string language)  
    {  
        return ((TimeSpan)value).TotalSeconds;  
    }  
}
```

5.3 File management 文件管理

媒体部分的文件管理，利用到了 fileOpenPicker，从外部打开文件:

```
private async void pickFileButton_Click(object sender, RoutedEventArgs e)  
{  
  
    // Create and open the file picker  
    FileOpenPicker openPicker = new FileOpenPicker();  
    openPicker.ViewMode = PickerViewMode.Thumbnail;  
    openPicker.SuggestedStartLocation = PickerLocationId.VideosLibrary;  
    openPicker.FileTypeFilter.Add(".mp3");  
    openPicker.FileTypeFilter.Add(".wma");  
    openPicker.FileTypeFilter.Add(".aac");  
    openPicker.FileTypeFilter.Add(".m4a");  
  
    StorageFile file = await openPicker.PickSingleFileAsync();  
    if (file != null)  
    {  
        var stream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read);  
  
        mediaPlayerElement.SetSource(stream, file.ContentType);  
        mediaPlayerElement.Play();  
    }  
}
```

5.4 Network accessing 网络通信

在首页加入天气查询:

```
<TextBlock TextWrapping="Wrap" Grid.Column="1" VerticalAlignment="Center" FontSize="15" Text="城市" Margin="0,0,4,0" HorizontalAlignment="Right">  
<TextBox x:Name="cityName" Grid.Column="2" TextWrapping="Wrap" VerticalAlignment="Center" Height="39" />  
<Button Content="提交" HorizontalAlignment="Left" Margin="5" Grid.Column="3" VerticalAlignment="Center" Click="Button_Click" FontSize="14" />
```

在后台完成网络通信进行天气查询:



```
/*-----网络访问: 获取天气部分-----*/
private void Button_Click(object sender, RoutedEventArgs e)
{
    weather.Text = "";
    temperature.Text = "";
    GetWeather(cityName.Text);
}

private async void GetWeather(string tel)
{
    try
    {
        // 创建一个HTTP client实例对象
        HttpClient httpClient = new HttpClient();

        // Add a user-agent header to the GET request.

        /*默认情况下, HttpClient对象不会将用户代理标头随 HTTP 请求一起发送到 Web 服务。
        某些 HTTP 服务器(包括某些 Microsoft Web 服务器)要求从客户端发送的 HTTP 请求附带用户代理标头。
        如果标头不存在, 则 HTTP 服务器返回错误。
        在 Windows.Web.Http.Headers 命名空间中使用类时, 需要添加用户代理标头。
        我们将该标头添加到 HttpClient.DefaultRequestHeaders 属性以避免这些错误。*/

        var headers = httpClient.DefaultRequestHeaders;

        // The safe way to add a header value is to use the TryParseAdd method and verify the return value is true,
        // especially if the header value is coming from user input.
        string header = "ie Mozilla/5.0 (Windows NT 6.2; WOW64; rv:25.0) Gecko/20100101 Firefox/25.0";
        if (!headers.UserAgent.TryParseAdd(header))
        {
            throw new Exception("Invalid header value: " + header);
        }
        // API 链接
        string getCityCode = "http://api.avatardata.cn/Weather/Query?key=e81ecb57345c46af99a9c74bfcd5d0b&cityname=" + cityName.Text;

        //发送GET请求
        HttpResponseMessage response = await httpClient.GetAsync(getCityCode);

        // 确保返回值为成功状态
        response.EnsureSuccessStatusCode();

        // 因为返回的字节流中含有中文, 传输过程中, 所以需要编码后才可以正常显示
        // "\u5e7f\u5dde" 表示“广州”, \u表示Unicode
        Byte[] getByte = await response.Content.ReadAsByteArrayAsync();

        // UTF-8是Unicode的实现方式之一。这里采用UTF-8进行编码, 为了保障中文的显示
        Encoding code = Encoding.GetEncoding("UTF-8");
        string result = code.GetString(getByte, 0, getByte.Length);

        JsonTextReader json1 = new JsonTextReader(new StringReader(result)); // 实例化json数据
        string flag = "";
        while (json1.Read()) // 读取json数据, 赋值给flag
        {
            flag += json1.Value;
        }
        int error1 = 0;
        int.TryParse(flag.IndexOf("error_code").ToString(), out error1);
        string error = flag.Substring(error1 + 10, 1);
        if (error == "0")
        {
            int m1 = 0, m2 = 0;
            int.TryParse(flag.IndexOf("info").ToString(), out m1);
            int.TryParse(flag.IndexOf("temperature").ToString(), out m2);
            //m2 = flag.IndexOf("mobiletype").ToString();
            string Weatherinfo = flag.Substring(m1 + 4, m2 - m1 - 4);

            int m3 = 0;
            int.TryParse(flag.IndexOf("dataUptime").ToString(), out m3);
            string Temperature = flag.Substring(m2 + 11, m3 - m2 - 11);
            weather.Text = Weatherinfo;
            temperature.Text = Temperature + "°C";
        }
        else
        {
            var i = new MessageDialog("");
            i.Content = "Please input the correct city name(using Chinese name)";
            await i.ShowAsync();
        }
    }
    catch (HttpRequestException ex1)
    {
        weather.Text = "出问题了哟, 请重新输入~";
        Debug.WriteLine(ex1.ToString());
    }
    catch (Exception ex2)
    {
        weather.Text = "请重新输入正确城市哦~";
        Debug.WriteLine(ex2.ToString());
    }
}
}
```

5.5 DataBinding 计划管理页面的数据绑定

在计划管理页面，日计划清单、月计划清单和年计划清单均用到了数据绑定：

```
<!--日计划清单-->
<Grid Grid.Column="1" Visibility="Visible" Name="dayPlanList">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="1*" />
    </Grid.ColumnDefinitions>
    <Grid Grid.Column="0">
        <StackPanel>
            <TextBlock Text="时间" HorizontalAlignment="Center" />
            <ListView Name="time">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind DayGoalString}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
        <!--在这里要使用模板-->
    </Grid>
    <Grid Grid.Column="1">
        <StackPanel>
            <TextBlock Text="计划完成事项" HorizontalAlignment="Center" />
            <ListView Name="name">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind name}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
        <!--在这里要使用模板-->
    </Grid>
    <Grid Grid.Column="2">
        <StackPanel>
            <TextBlock Text="完成时长" HorizontalAlignment="Center" />
            <ListView Name="totaltime">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind usedGoalString}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
        <!--在这里要使用模板-->
    </Grid>
</Grid>
```

```
<!--月计划清单-->
<Grid Grid.Column="1" Visibility="Collapsed" Name="monthPlanList">
    <Grid.RowDefinitions>
        <RowDefinition Height="1*" />
        <RowDefinition Height="1*" />
    </Grid.RowDefinitions>
    <Grid Grid.Row="0">
        <StackPanel>
            <TextBlock Text="待月计划列表" HorizontalAlignment="Center" />
            <ListView Name="monthPlan">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind name}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
    </Grid>
    <Grid Grid.Row="1">
        <StackPanel>
            <TextBlock Text="已完成月计划" HorizontalAlignment="Center" />
            <ListView Name="realMonthPlan">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind name}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
    </Grid>
</Grid>
```

```
<!-- 一年计划清单 -->
<Grid Grid.Column="1" Visibility="Collapsed" Name="yearPlanList">
    <Grid.RowDefinitions>
        <RowDefinition Height="1*" />
        <RowDefinition Height="1*" />
    </Grid.RowDefinitions>
    <Grid Grid.Row="0">
        <StackPanel>
            <TextBlock Text="待年计划列表" HorizontalAlignment="Center" />
            <ListView Name="yearPlan">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind name}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>

            <!-- 在这里要使用模板 -->
        </StackPanel>
    </Grid>
    <Grid Grid.Row="1">
        <StackPanel>
            <TextBlock Text="已完成年计划" HorizontalAlignment="Center" />
            <ListView Name="realYearPlan">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:MyGoalItem">
                        <UserControl>
                            <TextBlock Text="{x:Bind name}" HorizontalAlignment="Center" />
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>

            <!-- 在这里要使用模板 -->
        </StackPanel>
    </Grid>
</Grid>
</Grid>
```

在后台代码中，通过 ViewModel 的函数获取绑定的 ObservableCollection，从而将动态绑定完全实现：

```
/*-----通过动态绑定显示右侧的item-----*/
private void calendar_SelectedDatesChanged(CalendarView sender, CalendarViewSelectedDatesChangedEventArgs args)
{
    if (calendar.SelectedDates.Count == 0) return;
    DateTimeOffset sourceTime = calendar.SelectedDates[0];
    ObservableCollection<Models.MyGoalItem> items = myViewModel.findDayGoalCollection(calendar.SelectedDates[0].Year, calendar.SelectedDates[0].Time);
    itemsSource = items;
    name.ItemsSource = items;
    totalTime.ItemsSource = items;

    // 待完成的月计划
    ObservableCollection<Models.MyGoalItem> monthItems = myViewModel.findMonthGoalCollection(calendar.SelectedDates[0].Year, calendar.SelectedDates[0].Time);
    monthPlan.ItemsSource = monthItems;
    // 实际已完成的月计划
    ObservableCollection<Models.MyGoalItem> realMonthItems = myViewModel.findMonthGoalCollection(calendar.SelectedDates[0].Year, calendar.SelectedDates[0].Time);
    realMonthPlan.ItemsSource = realMonthItems;

    // 待完成的年计划
    ObservableCollection<Models.MyGoalItem> yearItems = myViewModel.findYearGoalCollection(calendar.SelectedDates[0].Year, calendar.SelectedDates[0].Time);
    yearPlan.ItemsSource = yearItems;
    // 实际完成的年计划
    ObservableCollection<Models.MyGoalItem> realYearItems = myViewModel.findYearGoalCollection(calendar.SelectedDates[0].Year, calendar.SelectedDates[0].Time);
    realYearPlan.ItemsSource = realYearItems;
}
```



```
/*-----查询-----*/  
// 查询: 按照year, month, day来查询日目标, 返回日目标item集合  
public ObservableCollection<Models.MyGoalItem> findDayGoalCollection(int year, int month, int day)  
{  
    ObservableCollection<Models.MyGoalItem> DayGoalCollection = new ObservableCollection<Models.MyGoalItem>();  
    for (int i = 0; i < this._allItems.Count; i++)  
    {  
        if (this._allItems[i].type == 0 && this._allItems[i].startYear == year &&  
            this._allItems[i].startMonth == month && this._allItems[i].startDay == day)  
        {  
            DayGoalCollection.Add(this._allItems[i]);  
        }  
    }  
    return DayGoalCollection;  
}  
// 查询: 按照year, month来查询月目标, 返回月目标item集合  
public ObservableCollection<Models.MyGoalItem> findMonthGoalCollection(int year, int month, int isFinished)  
{  
    ObservableCollection<Models.MyGoalItem> MonthGoalCollection = new ObservableCollection<Models.MyGoalItem>();  
    for (int i = 0; i < this._allItems.Count; i++)  
    {  
        if (this._allItems[i].type == 1 && this._allItems[i].startYear == year &&  
            this._allItems[i].startMonth == month && this._allItems[i].isFinished == isFinished)  
        {  
            MonthGoalCollection.Add(this._allItems[i]);  
        }  
    }  
    return MonthGoalCollection;  
}  
// 查询: 按照year来查询年目标, 返回年目标item集合  
public ObservableCollection<Models.MyGoalItem> findYearGoalCollection(int year, int month, int isFinished)  
{  
    ObservableCollection<Models.MyGoalItem> YearGoalCollection = new ObservableCollection<Models.MyGoalItem>();  
    for (int i = 0; i < this._allItems.Count; i++)  
    {  
        if (this._allItems[i].type == 2 && this._allItems[i].startYear == year  
            && this._allItems[i].isFinished == isFinished)  
        {  
            YearGoalCollection.Add(this._allItems[i]);  
        }  
    }  
    return YearGoalCollection;  
}
```

5.6.Calendar 控件的使用

在计划管理页面加入了 CalendarView 控件, 以及使用了它的各种官方事件和属性从而将计划页面的日历呈现出来:

```
<CalendarView Name="calendar" SelectionMode="Single" HorizontalAlignment="Center" DisplayMode="Month" SelectedDatesChanged="calendar_SelectedDates
```

5.7 磁贴



使用xml文件来设定磁贴格式

```
<?xml version="1.0" encoding="utf-8" ?>
<tile>
  <visual>

    <binding template="TileSmall" displayName="SmallGoal">
      <text hint-style="subtitle"></text>
      <text hint-style="captionSubtle"></text>
    </binding>

    <binding template="TileMedium" displayName="SmallGoal">
      <text hint-style="subtitle"></text>
      <text hint-style="captionSubtle"></text>
    </binding>

    <binding template="TileWide" displayName="SmallGoal">
      <text hint-style="subtitle"></text>
      <text hint-style="captionSubtle"></text>
    </binding>

  </visual>
</tile>
```

5.8.磁贴循环展示

在 OnNavigatedTo 函数中激活清空通知队列，触发磁贴更新事件

```
TileUpdateManager.CreateTileUpdaterForApplication().EnableNotificationQueue(true);
TileUpdateManager.CreateTileUpdaterForApplication().Clear();
this.cycle += new notificationqueueCycle(Add);
Add(this, EventArgs.Empty);

public delegate void notificationqueueCycle(object sender, EventArgs e);
public event notificationqueueCycle cycle;

private void Add(object sender, EventArgs e)
{
    Windows.UI.Notifications.TileUpdateManager.CreateTileUpdaterForApplication().EnableNotificationQueue(true);
    TileUpdateManager.CreateTileUpdaterForApplication().Clear();
    for (int i = 0; i < myViewModel.allItems.Count; i++)
    {
        string t = myViewModel.allItems[i].name;
        string d = myViewModel.allItems[i].note;
        UpdateTile(t, d);
    }
}

public void UpdateTile(string name, string note)
{
    XmlDocument xml = new XmlDocument();
    xml.LoadXml(File.ReadAllText("Tile.xml"));
    XmlNodeList texts = xml.GetElementsByTagName("text");
    for (int i = 0; i < texts.Count; i++)
    {
        ((XmlElement)texts[i]).InnerText = name;
        i++;
        ((XmlElement)texts[i]).InnerText = note;
    }
    TileNotification notification = new TileNotification(xml);
    TileUpdateManager.CreateTileUpdaterForApplication().Update(notification);
}
```

5.9.主页 DataBinding 部分

主页各个计划的计划名称，完成状况。

```
Text="{x:Bind name}"
```

```
IsChecked="{Binding isFinished, Converter={StaticResource checkboxconverter}, Mode=TwoWay}"
```

timePage 各计划名称和预计花费时间，已花费时间和还需付出时间。

```
Text="{x:Bind name}" VerticalAlignme  
Text="预计花费时间:" FontSize="12" V  
Text="{x:Bind totalGoalString}" For  
Text="已付出时间:" VerticalAlignmen  
Text="{x:Bind usedGoalString}" Horiz  
Text="还需要付出:" VerticalAlignmen  
Text="{x:Bind needGoalString}" FontS
```

5.10 Adaptive UI 部分

在 mainpage 提供了 Adaptive UI，能够根据页面的宽度变化，决定是否出现右侧的目标编辑：

```
<VisualStateManager.VisualStateGroups>  
  <VisualStateGroup x:Name="VisualStateGroup">  
    <VisualState x:Name="VisualState1">  
      <VisualState.Setters>  
        <Setter Target="goalEditPage.(UIElement.Visibility)" Value="Visible"/>  
        <Setter Target="mainPage.(Grid.ColumnSpan)" Value="1"/>  
      </VisualState.Setters>  
      <VisualState.StateTriggers>  
        <AdaptiveTrigger MinWindowWidth="700"/>  
      </VisualState.StateTriggers>  
    </VisualState>  
    <VisualState x:Name="VisualState0">  
      <VisualState.StateTriggers>  
        <AdaptiveTrigger MinWindowWidth="1"/>  
      </VisualState.StateTriggers>  
      <VisualState.Setters>  
        <Setter Target="goalEditPage.(UIElement.Visibility)" Value="Collapsed"/>  
        <Setter Target="mainPage.(Grid.ColumnSpan)" Value="2"/>  
      </VisualState.Setters>  
    </VisualState>  
  </VisualStateGroup>  
</VisualStateManager.VisualStateGroups>
```

5.11. App to app communication 部分

主页的分享按钮点击，实现了 app to app communication 的分享功能，将目标的具体信息进行分享：

```
/*-----app to app communication-----*/
private void share_Click(object sender, RoutedEventArgs e)
{
    d = e.OriginalSource;
    DataManager.ShowShareUI();
}

dynamic d;
// 邮件内容
private void DataRequested(DataTransferManager sender, DataRequestedEventArgs e)
{
    var item = d.DataContext;

    var defl = e.Request.GetDeferral();
    DataRequest request = e.Request;
    request.Data.Properties.Title = "我的目标";
    request.Data.Properties.Description = "目标: " + item.name;
    request.Data.SetText("目标: " + item.name +
        "\n开始时间: " + item.startTimeString +
        "\n结束时间: " + item.endTimeString +
        "\n预计花费时间: " + item.totalGoalString +
        "\n已付出时间: " + item.usedGoalString +
        "\n还需付出时间: " + item.needGoalString +
        "\n备注: " + item.note
    );
    defl.Complete();
}

// 直接跳转到计时界面
dynamic a;
private void count_button(object sender, RoutedEventArgs e)
{
    a = e.OriginalSource;
    ((App)App.Current).myViewModel.selectedItem = a.DataContext;
    Frame.Navigate(typeof(timePageDetail), "");
}
```

六、项目难点及解决方案

6.1 写model、viewmodel 的时候遇到一个问题：大家应该是在写好 model、viewmodel 后动工，这样他们才能够进行动态绑定等，于是，看着 UI 想着大家可能会需要一些什么数据，以及这些数据用什么类型来表示会比较方便他们使用也方便我进行数据库的保存加载，由于我们的项目数据较多，最后把这些都写好了，他们开始动工，但是他们发现给的并不是他们想要的，主要是数据类型，由于认为 int 可能会比较方便保存，同时比较方便进行创建删除 item 和



绑定，但是并不是想象的这样，实际上他们的动态绑定更需要 `string`，因此，在后面需要进行改动，以免再次改动，于是让她们自己写出一个她们的需求，他们需要从我这儿得到什么数据以及什么类型的，最后，我再统一修改了，最后解决了这个问题，从这个问题知道了，以后需要先需求再实现比较好；

6.2 在完成数据库加载存储这个功能时，遇到了一个小问题，数据库给的是 `int64`，而我们的 `int` 是 32 位的，但是由于自己已经写了很多 `int`，同时大家其他部分已经使用我的 `model` 进行工作了，所以取消 `int` 的使用不太好，于是就进行了一个 `int` 的转换，从这个我知道了，以后时间这些还是用 `string` 保存比较好，因为他们需要的是 `string` 然后我比较好保存的也是 `string`，我这次使用 `int` 反而是费了多的周折

6.3 页面设计

在设计主页的时候，一开始走入一个误区，很多元素的宽度和高度都被设定为一个具体的数值，导致在运行的时候发现，当拉伸缩小窗口的宽度的时候，出现了很多元素被遮挡等影响软件使用和美观性的现象。于是开始参考微软的一些案例，发现如果在设计的时候采用比例法来切割界面，可以很好的解决以上的问题。

6.4 `TimePage` 的数据绑定主要是各个计划的预计花费时间，已付出时间，还需付出时间这三个主要数据的绑定。一开始，我们的 `model` 设定的时间不是以 `datetime` 来存储，而是年月日时分分开存储，这样使得计算和绑定难度很大。然后组员向组长反映这一问题，一起探讨如何来存储这些数据比较方便前台的绑定显示，最终我们采用了这一方式：在每一个计划都添加三个 `string`，把预计花费时间，已付出时间，还需付出时间在计划形成的时候便计算出来，接着转化为 `string`，供应给前台的绑定。这样就解决了 `timepage` 的数据绑定问题。这一方法的难点在于，要确保后台数据的正确性。

6.5 难点主要有两个：一是在计时的时候离开该页面，再回来计时页面，如何判断应不应该继续计时；二是在计时过程中，还需付出时间一栏



的实时更新问题。

先说明我计时的实现方法。我先在具体目标时间详情页面（ `timePageDetail` ）

```
public sealed partial class timePageDetail : Page
{
```

加入布尔值 `isCounting` 来记录该页面是否在计时。点击计时按钮后会调用 `while()` 循环，以下是伪代码：

```
while (isCounting) {
    Count time; // 进行计时的工作
    Task.Delay(1000); // 延迟 1 秒
}
```

但我发现，离开该页面再回到该页面后，页面虽然也还在计时，但 `isCounting` 显示的似乎是“未在计时”。点击计时按钮，页面又调用了一次 `while()` 循环，导致1秒内已付出时间的秒数增加了 2。

使用该方法计时还存在另一问题：如何同时为多个目标进行计时。

后来我在 `Model` 中

```
public class MyGoalItem : INotifyPropertyChanged
{
```

加入变量来记录该目标是否在计时中：`private bool _isCountingTime`。每次点击计时按钮，判断该目标是否处于计时状态：

```
while (goalItem.isCountingTime) { // goalItem 即 ViewModel.selectedItem (下同)

    Count time; // 进行计时工作

    Task.Delay(1000); // 延迟一秒

}
```

这样，假设用户点击了计时按钮并离开该页面。若是再进入同一个目标的计时页面，`while` 循环还在进行着，`goalItem.isCountingTime` 为 `true`，用户点击计时按钮，会暂停该计时；若用户进入另一个目标并点击计时按钮，因为 `goalItem.isCountingTime` 为 `false`，则创建一个新的 `while` 循环，为目标计时。解决了不同目标同时计时的冲突。

第二个问题中，我起初是用 `TextBlock` 来绑定 `goalItem.needGoalString`（还需付出时间）的：

```
<TextBlock Text=" {x:Bind goalItem.needGoalString, Mode=OneWay}" />
```

但发现，在计时的过程中，TextBlock 却没有实时更新。于是改变显示的方式：

xaml 文件中：

```
<TextBlock x:Name="TimeNeedToSpend" Margin="8,0" Grid.Column="1"/>
```

cs 文件中：

```
while (goalItem.isCountingTime)

    { Count time; // 进行计时工作

    Update goalItem.needGoalString; // 更新还需付出时间要显示的内容

    TimeNeedToSpend.Text = goalItem.needGoalString; Task.Delay(1000);

    // 延迟一秒

    }
```

这样，就完成了还需付出时间一栏内容的实时更新