计算机科学与工程学院课程设计报告

题目全称： 英语单词本

题目难度等级： 3

指导老师： 郭迅 **职称：** 高级工程师

序号	学生姓名	学号	班号	成绩
1	陈小羽	2016060106020	2016060106	
2				
3				
4				
5				

（注：学生姓名填写按学生对该课程设计的贡献及工作量由高到底排列，分数按排名依次递减。序号排位为“1”的学生成绩最高，排位为“5”的学生成绩最低。）

主要任务：实现英语单词本

1 可录入单词并分级。

2 随机背诵所有单词，

3 随机背诵某级的单词。

4 背诵分为中文背英文，英文背诵中文。

详细功能描述：

通过 FLUTTER 框架，通过一份代码，实现了可在 Android 和 IOS 两大平台发布的单词本应用程序.抛弃了传统单词本应用的模式，实现了一个简洁，易用，美观的新型单词本. 这个单词本可以再用户想要学习某个单词的时候，自动帮助用户在网上收集相关资料(包括百科，图片，意思，读音)并整理到应用中，方便用户随时记忆. 应用中还内置了多个单词列表，使得用户可以根据自己的习惯来整理相应的单词. 为了能够吸引用户使用，该应用中还有大量复杂的交互动画. 在 FLUTTER 框架强大的性能的加持之下，应用在智能手机上能够流畅顺滑的运行，交互体验十分良好.

预期成果或目标：

软件

指导老师评语：

指导教师签字： _____

第1章　需求分析

最主要的需求是：制作一个单词本程序.

1.1　市面上的其他单词本程序

要制作一款优秀的单词本应用，首先去应该去解了一下市面上都有哪些单词本应用．在进行了一些调查之后可以发现几乎没有桌面端的类似应用的影子，这一点倒也容易想到．因为一般用户背单词一般都使用碎片时间，很少会有人坐在电脑面前背单词．所以我最终决定要做一个移动端的单词本，这样才能够把这个应用真的做成和一个能够随身携带小本本一样，使得用户可以充分利用他的碎片时间来记忆单词.

那么那些市面上的移动端的单词本都是怎么样的呢？下面是现在比较知名的移动端应用.

1.1.1　扇贝单词

FIGURE 1 扇贝官网对自己应用的介绍

扇贝单词这个应用提供了很多的单词书．用户可以订阅很多本单词书．然后这个应用可以在每天引导和提醒用户记忆这些订阅之后的单词书上面的单词．在背单词的时候，这个应用会不停的出现用户见过或者没有见过的单词，如果用户能够想起这些单词的意思，那么新的单词就会出现，如果用户没有回忆起来这些单词的意思，那么这些单词就会在之后的某个时刻再次跟用户见面．很容易发现，扇贝英语这个软件通过其大量的单词书，和自动化的背单词机制，为广大的学生群体提供了背单词的有力工具.

1.1.2　百词斩

FIGURE 2 百词斩官网介绍

百词斩这个应用和扇贝很像，他们主打的方式都是通过一些在记忆领域的研究结果，帮助用户总结了一套自动化的背单词流程，结合强大的单词库，使得对考试有需要的用户可以通过这些应用在短期内记忆较为大量的单词，从而应付一些考试．百词斩的单词没有扇贝那么多，所以他们别出心裁，推出了通过图像来记忆单词的机制．就是说，在百词斩中出现的每个单词都配有一张图，通过理解这张图的内容，用户可以更快的将中文意思和英文意思产生联系，从而避免枯燥乏味的记忆过程．而且这样记忆之后并不那么容易忘记，看到图片之后又会很快的想起来．

1.2 要做一个怎样的单词本

上面两个单词本软件都很优秀．但是他们都只关注考试，他们所推崇的记忆方式也都是针对考试的，说实话，这样的记忆方式(即使有图)也是很难长久的．因为用户没有真正的使用过单词，没有在自己阅读的文章中看到过这个单词，没有去掌握这个单词使用的语境．这显然违背了人脑认知世界的方式，形式化的东西在脑袋里面呆的时间总是很短的．

所以，我最终决定做一个没有单词库的单词本，即单词本需要用户手动添加单词．不过手动添加单词是一个很麻烦的工作．如果需要用户手动添加这个单词的意思和用法等信息的话，估计这个软件就没有人用了.所以希望用户只需要输入他想添加的单词，然后程序就会自动地去一些比较权威的网站上找到该单词的意思和用法，然后呈现在用户面前．这样，用户在生活实际或者阅读过程中，如果遇到了比较感兴趣的单词，便可以直接在应用中添加该单词．然后该单词相关的意思，百科，图片就会自动的被收集到用户的手机中，供用户现在和以后查看．这种学习方式适合长期学习的用户，虽然对短期的考试可能没有什么效果，但是如果想要提升英语水平的话，这其实是一种非常有效的方式．

在确定了单词本的主要功能之后，我希望单词本能够融入到用户的生活之中．使得用户在使用单词本的过程中不会有任何痛苦的感觉，只有享受的感觉．所以界面一定要足够的简洁，使用起来要足够的简单，使得用户能够没有任何负担的使用该程序．

第2章 开发平台和框架

在有了对整个项目的大致定位和想法之后. 首先遇到的问题便是要选择什么平台和框架来进行开发. 因为我想要开发移动端的应用, 而安卓和 IOS 都各有一套自己的开发平台和方案, 所以这一度成为了一个棘手的问题. 我希望做的是一个跨平台的应用, 即在任何主流的手机上应该都要可以运行才好. 在收集了大量资料过后, 我最终选择了 Google 近两年推出的 Flutter 开发框架开实现单词本.

2.1 FLUTTER 框架

FLUTTER 是 Google 开发的一个新一代的开源应用开发框架. 它官网的宣传标语就是 Beautiful native apps in record time. 指在短时间内开发出令人惊艳的原生应用. FLUTTER 框架通过利用 Google 领导的下一代网页排版语言 Dart 作为开发语言, 再由其内置的编译器将 Dart 描述的逻辑功能转化为 Android 或者 IOS 设备上能够运行的原生代码. 这样只用写一份代码就能在两大平台上运行该的应用. 而且 FLUTTER 应用在运行时使用自绘引擎, 可以充分利用手机 GPU 的能力, 这使得通过 FLUTTER 实现的应用程序甚至可以在好几年前的机型上实时绘制出 60Hz 刷新率的动画效果.

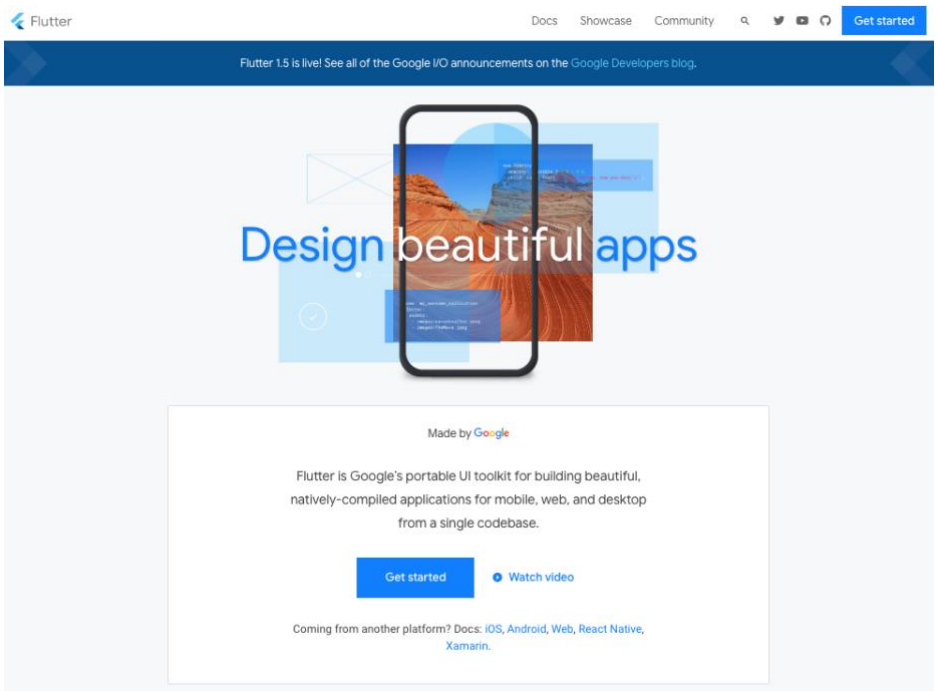


FIGURE 3 FLUTTER 官网的一些介绍

2.2 FLUTTER 环境配置及使用方法

可使用在官网给出(https://flutter.dev/docs/get-started/install)的安装方法来配置开发环境. FLUTTER 官网还有非常细致的用法示例和教学(https://flutter.dev/docs).

第3章 顶层设计

3.1 视觉效果设计

受用户欢迎的应用程序都有一个共同点, 那就是好看. 一个好看的界面对于吸引用户使用具有十分重要的作用. 为了便于模块化, 我将应用的界面分为了很多个小的部分, 下面来一一介绍这些小部件.

3.1.1 视口设计

视口是整个应用的界面的最上层，这一层主要集中的是一些用户常用的按钮，单词本切换等功能．视口的下面会放一些可以滑动的列表，或者单词卡片之类的动态的东西．相对于用户来说，视口从程序开始运行到程序结束退出一直都是和用户相对静止的．

FIGURE 4 视口设计效果

整个视口的上方都是非常干净整洁的，没有一丝多余的东西．视口的最下方是一个 TabBar 用于显示用户现目前处在哪个级别的单词上．通过点击 TabBar 或者在视口上空白的区域滑动手指，用户可以切换当前所在的单词本．视口上还有一个紫色的➕号按钮．这个按钮负责实现添加新单词的功能，在 FLUTTER 的术语中，这个按钮被称为 floatingActionButton．意思是它浮动在了视口留出的空白区域的上方．至于这个➕号按钮按下去之后的效果和这个圆润的视口是如何设计并实现的，我把他们留在细节设计讨论．

3.1.2 单词卡片

我在视口中，通过卡片的方式来呈现单词．这样给人一种简洁的美感．通过将单词卡片放到一个滑动列表的上方，用户可以滑动单词卡片，形成一种流畅的感觉．

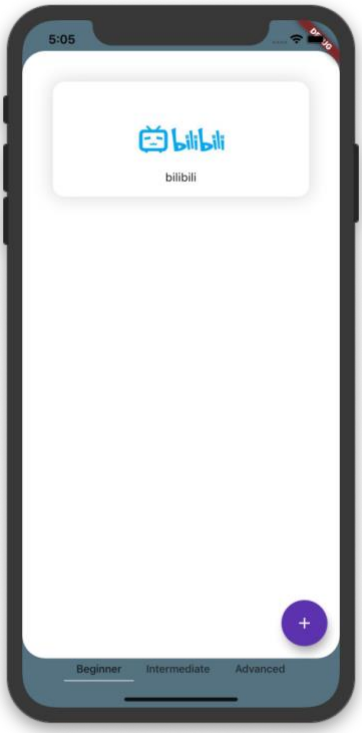


FIGURE 5 单词卡片的视觉效果

3.1.3 单词详情页面

当用户点击某个单词卡片时，需要能够将其展开成为一个单独的页面．通过这个单独的页面，可以给用户展示这个单词的意思，同义词，图片，百科，读音．当用户再次点击图片，这个页面又能够收起来，还原为一个单词卡片．

FIGURE 6 单词详情页面

3.2 数据库设计

大多数应用程序在设计的过程中，都需要设计相应的数据结构来保存应用的状态. 在实现的这个应用中，最主要的需求时记录用户之前添加过那些单词，这些单词的意思以及一些其他的属性，还要将这些单词和一些特殊的单词表关联起来.

所以，我为应用设计了一个简单的数据库. 首先为数据库绘制了 E-R 图.

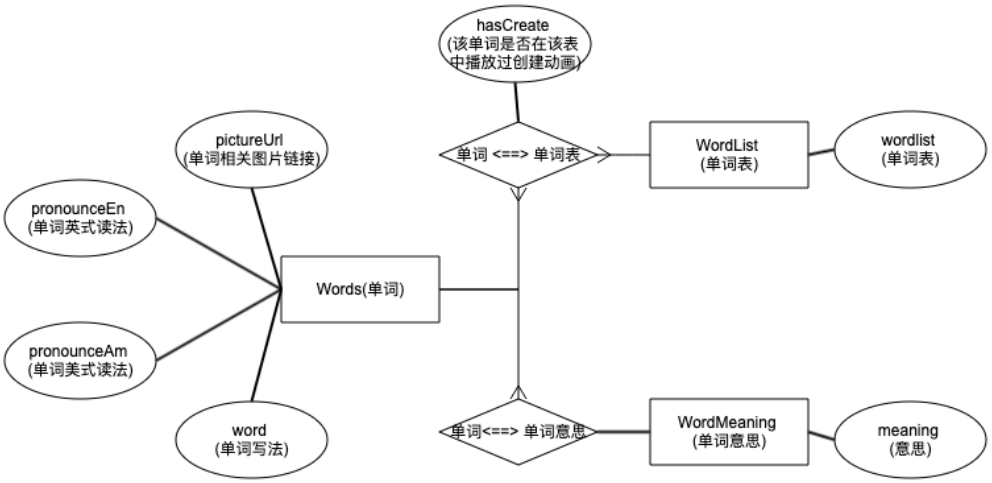


FIGURE 7 ER-图

从图中可以看出，单词表这个实体和单词意思这个两个实体都只有一个属性，所以单词表实体实际上可以和单词⇔单词表关系合并成一个表，同理单词意思实体可以和单词⇔单词意思合并成为一个表. 最终得到的表结构通过 SQL 语句表示如下:

```sql
CREATE TABLE Words (
    word varchar(100) primary key,
    pronounceEn varchar(100),
    pronounceAm varchar(100),
    pictureUrl varchar(10000)
);
CREATE TABLE WordMeaning (
    word varchar(100) not null,
    meaning varchar (10000) not null,
    primary key (word, meaning),
    foreign key (word) references Words(word) on delete cascade
);
CREATE TABLE WordList (
    word varchar (100) not null,
    wordlist varchar (100) not null,
    hasCreated boolean DEFAULT false,
    primary key(word, wordlist),
    foreign key (word) references Words(word) on delete cascade
);
```

FIGURE 8 数据库建立

3.3 工作机制设计

因为按照之前的想法， 单词本需要能够通过用户输入的单词自动到网上收集关于这个单词的各种意思，图片，发音. 而这些工作需要用到大量的网络连接. 总所周知，网络传输是非常缓慢的，可能过了很久都没把需要的信息获取完全. 而且在获取信息的过程中不能够阻塞进程，还得继续和用户交互(画面不能卡住).

为了能够完成这样的需求，我设计了一个支持异步的应用工作机制.

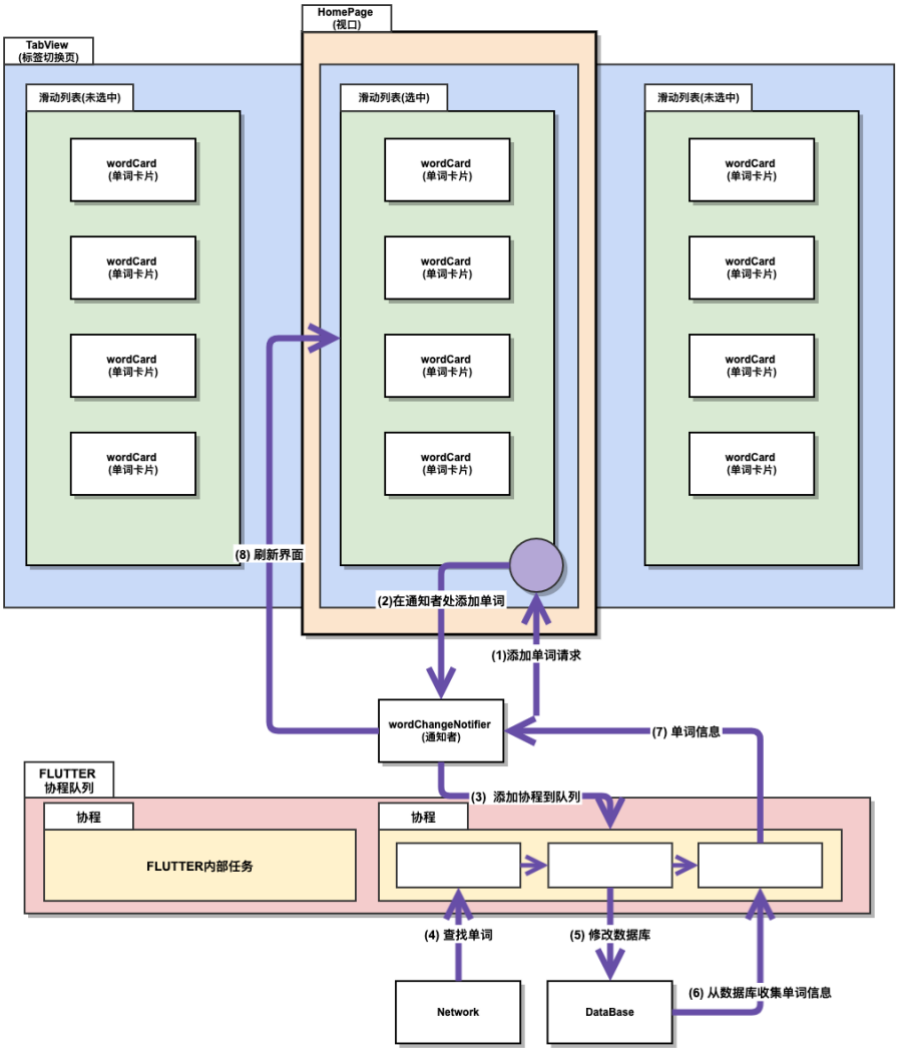


FIGURE 9 异步机制

我利用了 FLUTTER 的协程机制来实现异步添加单词. 当用户发出了添加单词的请求之后，会将整个搜索资源和修改数据库的操作打包成为一个协程，然后将这个协程添加到 FLUTTER 的协程队列中. 协程机制十分有趣，整个协程队列中的协程会循环的被执行，但是和一般的函数执行不同的是，一个协程只要发现自己被阻塞了，就可以主动挂起自己，让自己之后的协程继续执行，从而不会阻塞整个应用.

FLUTTER 的协程机制中，只要发生异常，整个协程就会被从协程队列中清除出去. 该协程之后剩下的代码不会执行. 思考一下之后可以发现，这个过程唯一可能对代码的鲁棒性有影响的地方在于，有可能数据库在于，有可能在数据库修改到一半的时候发生异常. 不过仔细观察(5)(6)两个步骤可以发现，当数据库更新失败时，协程不会继续运行，即不会去刷新界面，所以界面会一直和数据库中的内容保持同步. 而且现代化的数据库系统都有事务回滚这种机制来防止更新错误，所以数据库中的数据也不会出错. 综上，这个机制即不会阻塞画面和用户交互. 而且，它执行的结果只有两种, (1) 成功，这种情况下皆大欢喜 (2)失败(可能由于各种原因，比如网络连接不稳定)，这种情况下也不会影响程序的正确性和程序的后续执行，这次添加单词的请求就像没发生过一样.

删除单词和这个流程非常几乎一样，只是没有第(4)步而已，这里就不再赘述了.

3.4 小结

至此，对整个项目的大致规划已经完成. 通俗来说，在这一小节中，主要探讨了以下几个问题:

1. 本应用的定位.
2. 本应用主要应该提供的功能.
3. 本应用应该长什么样.
4. 本应用中，需要保存那些数据.
5. 需要在不阻塞界面的情况下从网络获取资源以及更新数据库，数据更新完成之后还需要刷新界面，出错之后还要保证程序能够正确运行，要通过什么样的机制来实现这么多的需求.

对于最棘手的异步问题，也已经大致解决. 但是其实要完整的实现该应用还有很多问题需要考虑，比如:

1. 用户具体如何添加单词.
2. 单词卡片和单词详情页面应该用怎样的动画切换.
3. 如何从网络上收集某个单词的相关信息.
4. 如何在代码中组织数据.

下面的篇幅要开始在这些具体的细节问题上进行展开.

第4章 细节设计

4.1 从网络上获取单词信息

目前很多网站都有很多反爬虫机制，有些网站上虽然有用户想要的信息，但是直接从移动端获取这些信息却不太容易. 特别是图片的链接，很多网站对图片链接有非常严格的保护. 有的通过 JS 代码，通过懒加载的方式获取图片链接；有的干脆直接没有图片链接，通过 JS 代码传递二进制序列化之后的图片，然后通过 JS 重新在网站上绘制出来.

不过，经过一些探索，终于还是在网络上发现了两个不太设防的网站. 一个是有道词典网页版，可以从这个网站获取单词的意思和读音等信息. 还有一个是互动百科，可以从这个网站上获取单词相关的图片和一些单词的背景知识(不过主要还是图片).

4.1.1 在 FLUTTER 中使用 HTTP 协议

要想和特定的网站进行交互，必须能够在应用程序内部运行 HTTP 协议. 这样才能获得这个网站的页面上的 HTML 代码，从而找到需要的资源. FLUTTER 中有一个 package dio 可以很好的解决这个问题. 比如，当想抓取有道词典网页版中关于某个单词的页面的时候，可以这样写:

```
// get document from internet
// @param: String keyword, access the youdao dictionary using keyword
// @ret Future<Document>, the document represent the web page
Future<Document> getDocument(String keyword) async {
  var dio = Dio();
  Response rsp = await dio.get(
    "https://www.youdao.com/w/eng/" + keyword +"/",
    options: Options(responseType: ResponseType.json)
  );
  var document = parse(rsp.data);
  return document;
}
```

FIGURE 10 DIO 用法

可以看到, dio 提供了一个异步的 get 函数，支持通过 http 协议中定义的 get 方式访问特定的域名，并在完成访问之后将信息返回 (调用该方法的函数也必须被标记为异步，即协程).

4.1.2 从 HTML 页面上精确的获取目标信息

获取到特定页面的 HTML 代码过后，需要从这个页面上精确的获取想要的信息. 目前已知的，有两种方法.

4.1.2.1 生成 DOM 树并在其上利用元素选择器来进行 HTML 标签的查询

FLUTTER 提供了一个 package html 来将纯文本的 HTML 代码通过词法和语法分析翻译为一棵结构化的 DOM 树.通过这个功能，再配合上 package html 提供的元素选择器功能,就可以精确的找到网页文件中满足某些特征的标签，进而从这个标签当中获取需要的信息. 比如在有道词典网页版中查找单词的意思的时候，就使用了这种方法:

```
void findEnglishMeaning(Document document, Map<String, dynamic> word) {
  List<Element> list = document.querySelectorAll('.trans-container .wordGroup .search-
js');
  if (!(word['meaning-list'] is List<String>)) {
    word['meaning-list'] = new List<String>();
  }
  for (final e in list) {
    word['meaning-list'].add(e.innerHtml);
    print(e.innerHtml);
  }
}
```

FIGURE 11 用于获取单词意思的代码

4.1.2.2 直接通过正则表达式抓取特征串

Dart 这种语言内置了正则表达式匹配的功能. 所以也可以利用正则表达式来获取想要的信息. 比如为了获取互动百科上的图片，我设计了这样的正则表达式:

```
RegExp exp = RegExp(r'(<div\s*style\s*=\s*"margin:0\s*auto;\s*display:none;\s*">\s*
  <img\s*src\s*=\s*"[^\"]*"\s*/?>)');
```

FIGURE 12 用于抓取图片的正则表达式

4.2 通过 HERO 动画切换单词卡片和单词详情页

其实在 FLUTTER 的机制中，单词卡片和单词详情页是处于于两个不同页面中的不同的元素. 所以在这两个东西之间的切换相当于在 FLUTTER 中进行了一次换页操作. 在 FLUTTER 中，页是 UI 的根结点，所以换页操作相当于需要重新绘制另外一个完全不一样的 UI 界面. 这样就导致在这两个元素之间切换非常难做到平滑. 但是这个地方非常决定用户的体验，所以我决定着手解决这个问题. 目前发现的一个比较优秀的解决方法是使用 HERO 动画，这种转场的方式被广泛的使用在很多手机应用当中.

4.2.1 什么是 HERO 动画

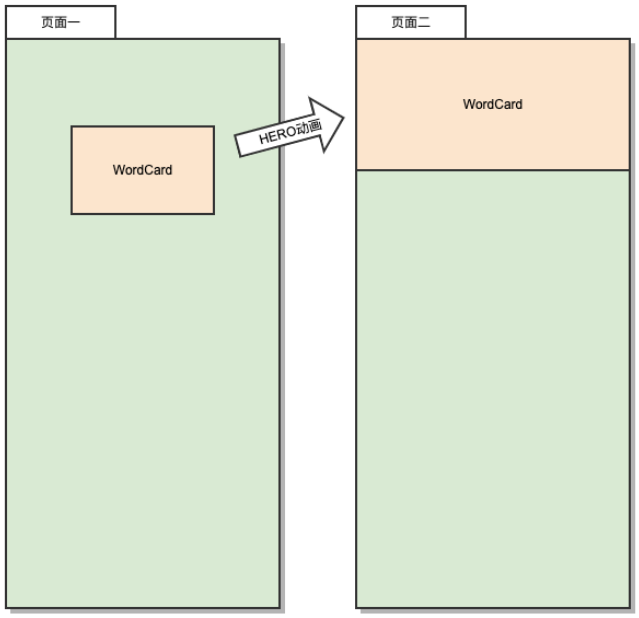


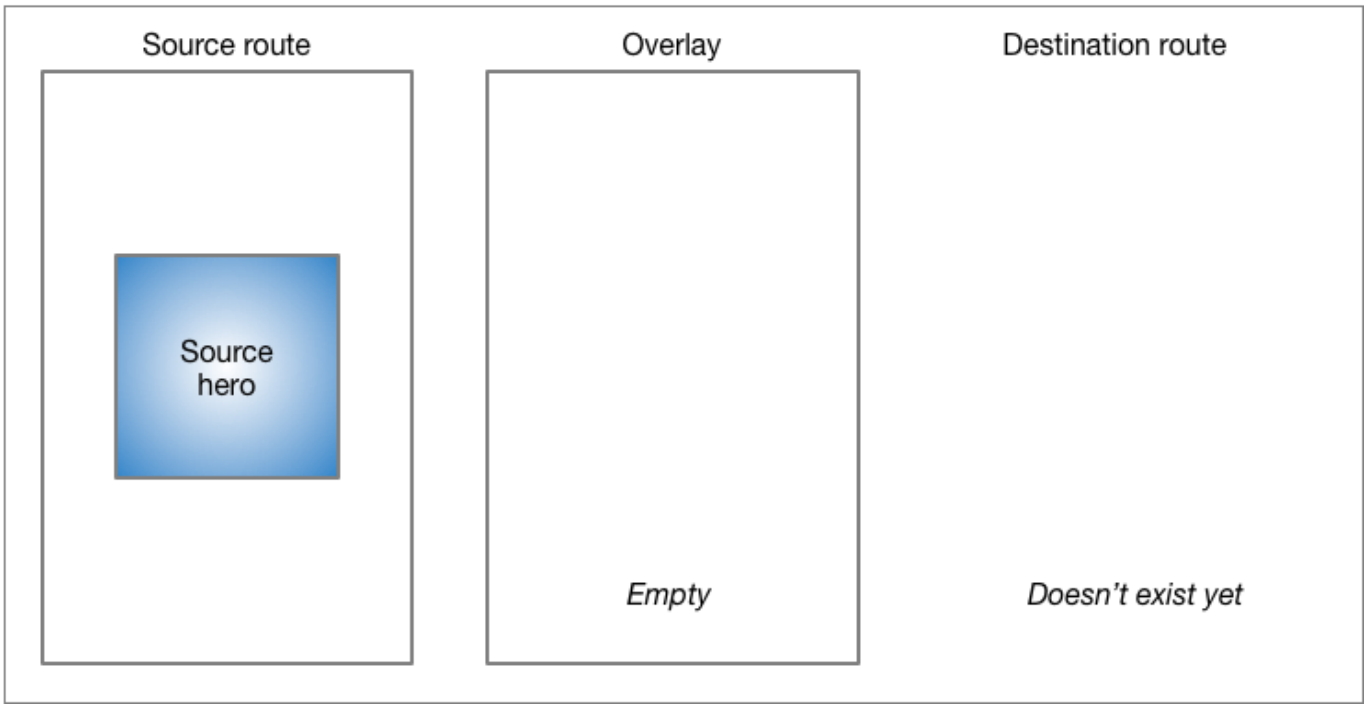
FIGURE 13 HERO 动画示意图

在切页时，如果两个切换的页面之间恰好有公共元素，则可以使用 hero 动画. Hero 动画通过在切页时，强调两个页面的公共元素来实现无缝切换的效果. 例如，在 Figure 13 中的 hero 动画播放时，用户只会看到 WordCard 对应的区域向上飞并且在宽度方向充满整个屏幕，并不会感觉到已经切页了. 但实际上已经切页了. Hero 动画其实是一个来自于魔术的手法，通过 Hero 元素来吸引用户的注意力，从而让用户难以注意到切页中的不自然的地方.

4.2.2 如何实现 HERO 动画

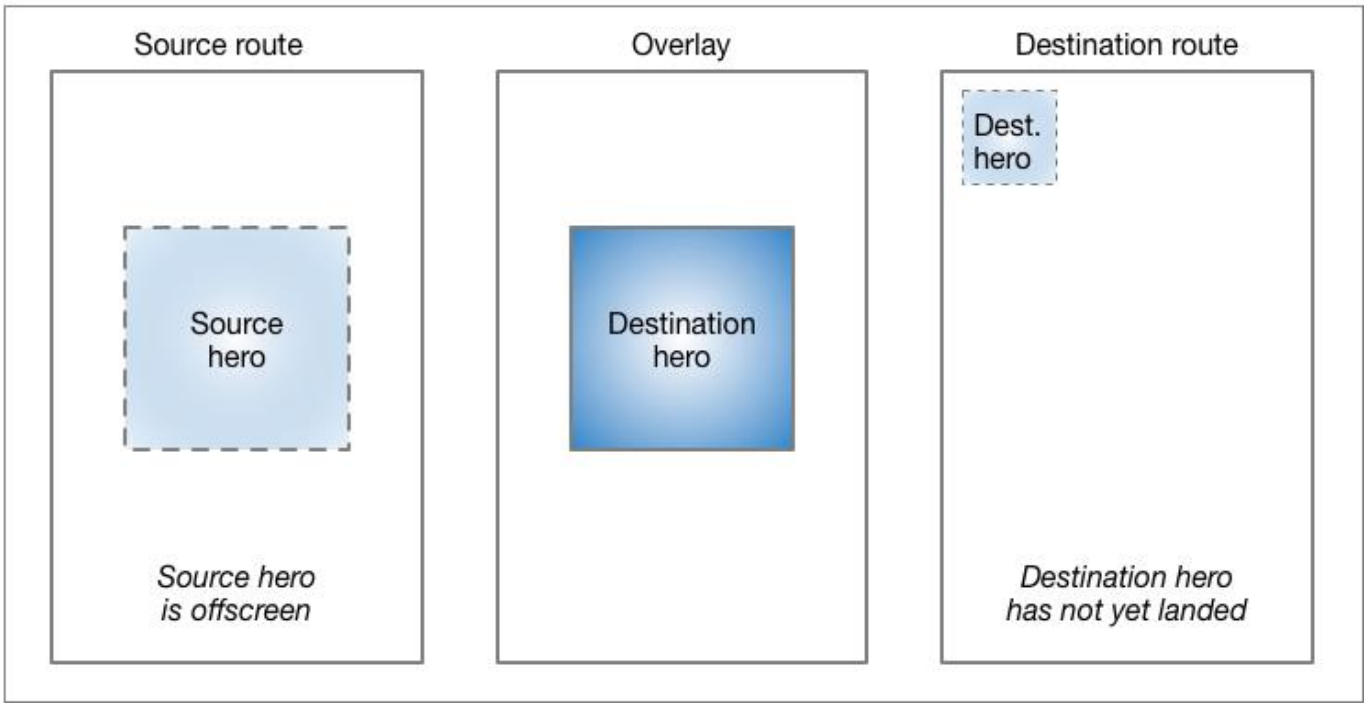
这么厉害的动画，如何实现呢？ Flutter 官方不仅提供了 Hero 动画这一概念，还教了应该如何实现 Hero 动画. 实现 Hero 动画主要分四个步骤，下面来稍微介绍一下他们.

0) Before transition

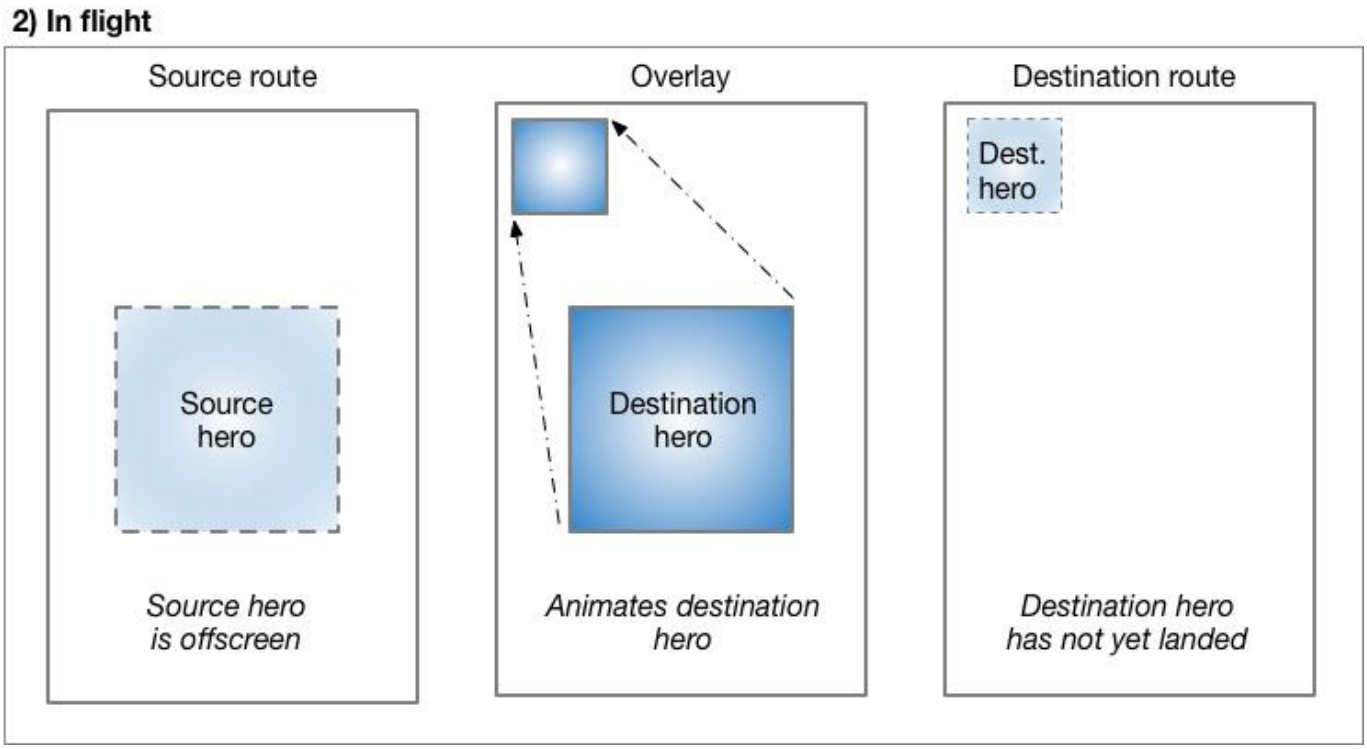


刚开始时，目标页面还没有被加载出来.

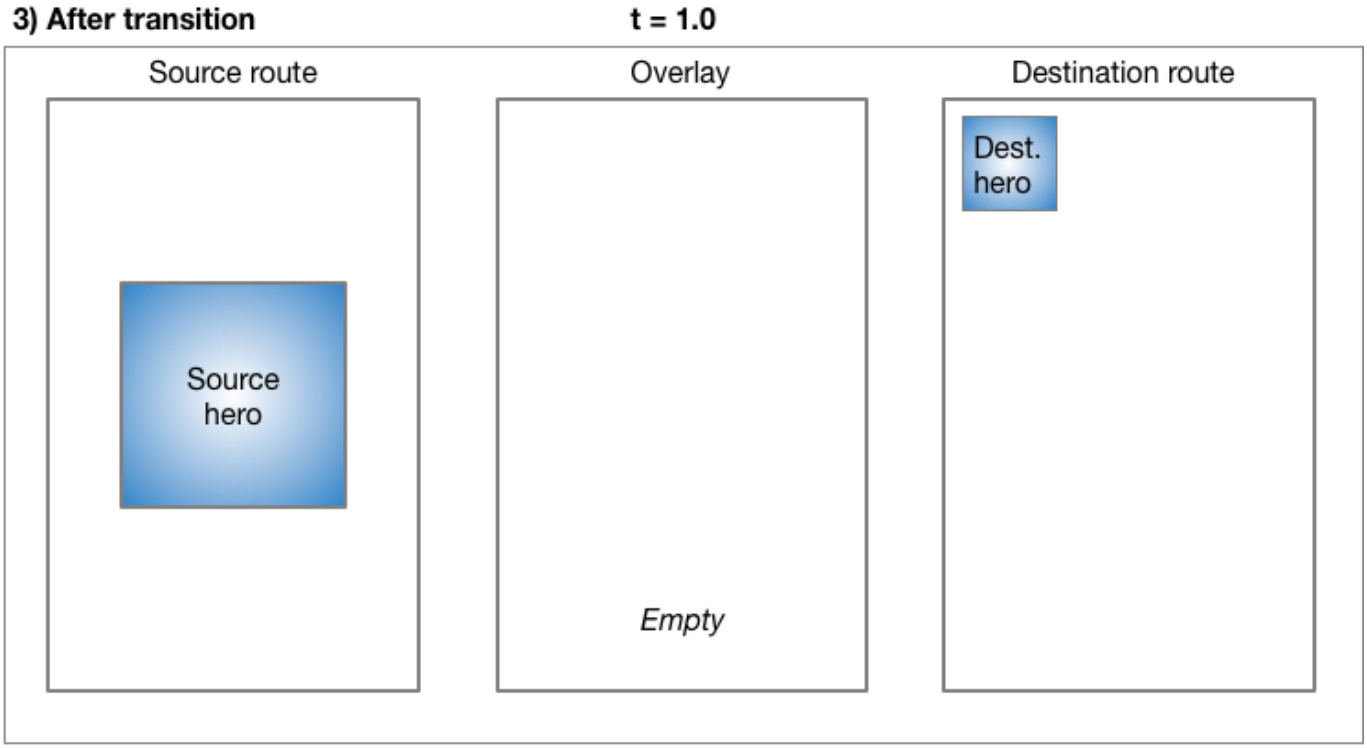
1) Transition begins when destination route is pushed to Navigator **t = 0.0**



Hero 动画开始时，当前页面和目标页面同时被加载出来，不过这个时候两个页面显示的透明度之类的属性不太一样. 这个时候最重要的一点是，两个页面的 Hero 元素(看起来像，但实际上不是同一个元素)会被通过形变和位移之后重合在一起. 初始时的位置在当前页面的 Hero 元素的位置上.



然后开始播放一个飞去的动画，两个页面上的 Hero 元素同时往目标页面的 Hero 元素的位置飞.



动画播放完成之后，销毁第一个页面.

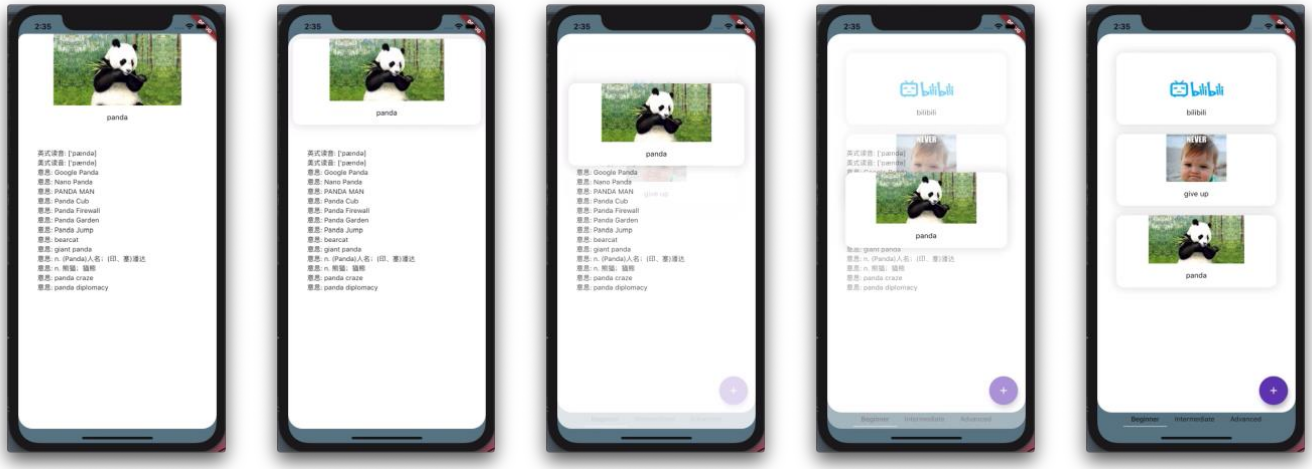
其实 FLUTTER 已经做好了 Hero 动画的大部分规定动作. 但是为了能够将 FLUTTER 的 Hero 动画调整为需要的形式，所以需要对 Hero 动画的各个环节做一个大致的了解.

4.2.3 如何将 FLUTTER 提供的 HERO 动画调整为需要的形式

FLUTTER 提供的原生 HERO 动画只实现了 Hero 元素飞的效果，但是其他元素的切换依旧是非常突兀的. 我们需要在吸收 FLUTTER 维护的一部分动画的同时，加一些自己的动画，这样动画才会比较平滑和自然. 这里，通过继承 StatefulWidget 这个基础类型创建了 WordCard 类型，这使得可以为 WordCard 添加自定义的动画. 接着,在播放 Hero 动画的同时通过时间为参数，不停的在每一帧调整两个页面的透明度. 这样就实现了非常平滑的过度. 具体实现时要注意的细节还是非常多的，这里就不展开说了.

4.2.4 HERO 动画的最终效果

下面这些图从左到右，分别表示了动画过程中的一些状态.



4.3 单词卡片的生命周期

添加和删除单词需要访问网络并对数据库进行操作. 最后，当数据更新完成之后，再刷新界面. 访问网络和更新数据库的操作在前面的部分都提到了，这里对刷新界面的相关工作进行说明. 为了能够使得卡片在出现的时候不是特别的突兀，我在新建卡片时和销毁卡片时分别制作了动画.

对于新建卡片时播放的动画，卡片初始的 height 属性为 0，这个时候即使卡片突然出现在了队列中，用户也是看不到的，之后在播放动画的过程中，卡片的 height 属性再慢慢变为正常值.

对于销毁卡片时播放的动画，卡片的初始 height 属性为正常值，播放动画的过程中，卡片的 height 属性变为 0，在 height 属性变为 0 的瞬间，触发事件将整个卡片对象从队列中拿掉，用户就看不出来了.

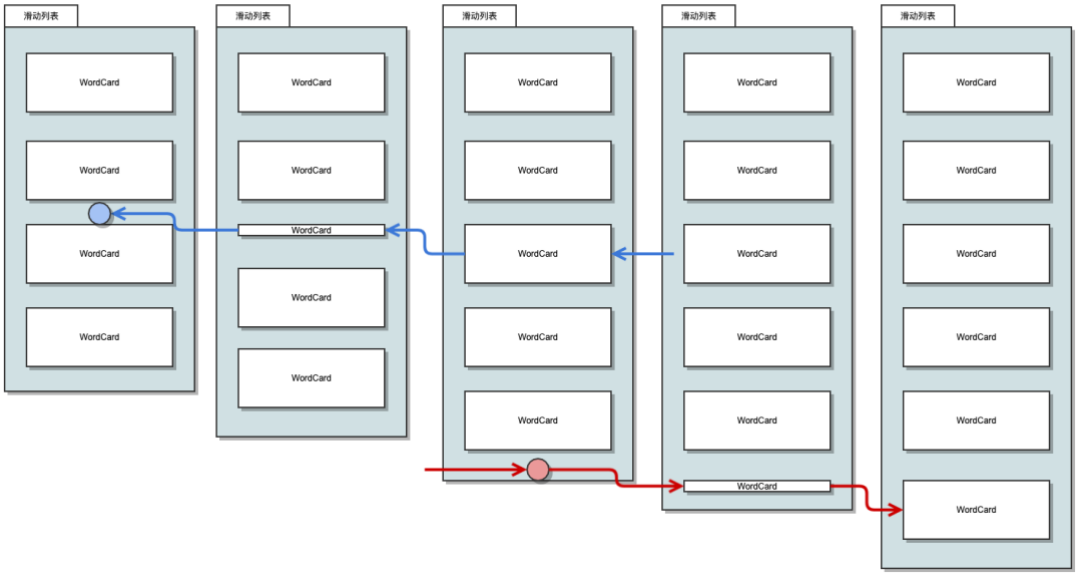


FIGURE 14 添加删除卡片示意图

4.3.1 卡片创建和销毁动画的制作

FLUTTER 提供了「动画控制器」这样一个工具.可以通过这个工具来制作自定义动画. FLUTTER 中，动画控制器是一个能按照时间顺序输出 0 → 1之间数字的信号发生器，另外它还支持:

1. 从0 → 1播放 / 从1 → 0播放.
2. 添加事件监听器，在信号到达某些状态时给监听器发信号.

动画控制器中，常用的监听器有 Listener 和 StatusListener. Listener 只要信号发生改变就会向外通知，而 StatusListener 只会在动画控制器的逻辑状态(播放中，停止，反转)发生改变时才会向外通知. 通常可以使用 Listener 来通知 FLUTTER 重绘界面，而使用 StatusListener 来触发某些逻辑操作.

比如在删除卡片时，当检测到卡片缩小的动画播放完之后，会通知数据库清除当前卡片相关的所有数据，这样这个卡片才算是真正的销毁掉了.

在新建卡片时，在数据库中维护了该卡片是否已经在该列表中播放过新建动画，如果已经播放过，则本次不用播放，否则将给动画控制器设定上特定的初始之然后播放新建动画. 新建动画播放完成之后，对应的 StatusListener 会通知数据库将关于动画是否播放过的表项给修改为播放过.

4.3.2 删除单词的相关操作

添加单词时通过✚号按钮来实现，因为✚号按钮内部过于复杂，所以在后面专门用了一个小结来讲解它的实现. 所以这个地方主要来讲解一些删除单词是如何实现的.

在该应用中，没有专门制作删除单词的按钮，而是选择通过长按这种手势操作来进行单词的删除. 通过 FLUTTER 的 GestureDetector 检测长按事件，长按时间发生之后，在整个节目的底部显示一个小弹窗，这个小弹窗是 FLUTTER 提供的标准功能. 该应用在小弹窗上询问用户是否确定删除这个单词，用户点击确定之后，该应用会在添加删除单词的 StatusListener 之后开始播放删除单词的动画. 如果用户很长时间都没有点击确定按钮，小弹窗会自行消失.

这个功能的通过代码是这样实现的:

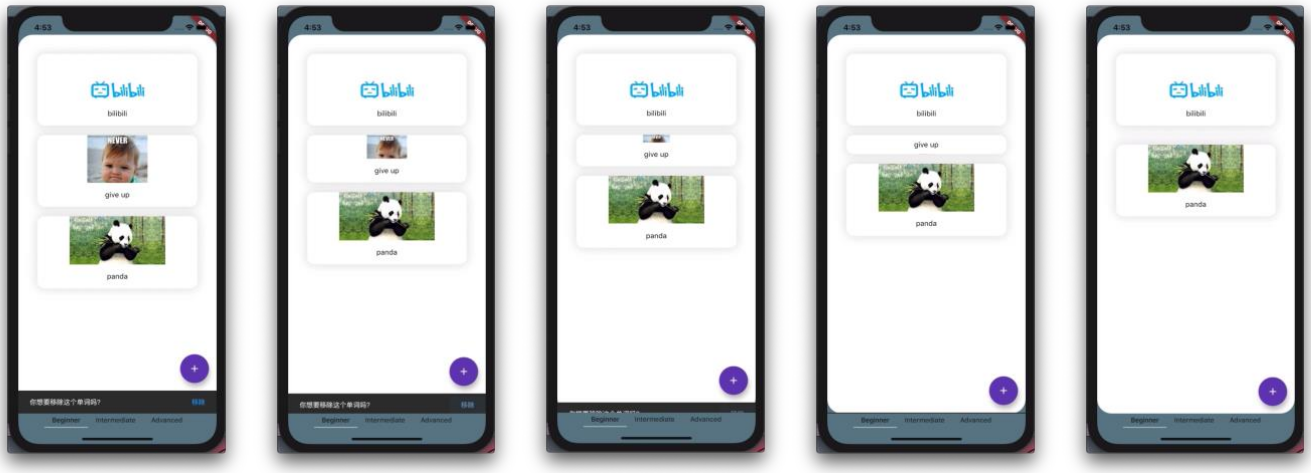
```dart
Scaffold.of(context).showSnackBar(
  SnackBar(
    content: Text('你想要移除这个单词吗?'),
    action: SnackBarAction(
      label: '移除',
      onPressed: () {
        controller.addStatusListener((status) {
          if (status == AnimationStatus.dismissed) {
            Provider.of<WordChangeNotifier>(context)
                    .removeWord(widget.word['keyword']);
          }
        });
        controller.reverse();
      },
    ) ,
  )
);
```

FIGURE 15 删除单词所用的代码

其中, WordChangeNotifier 类型负责修改数据库，并在完成之后负责通知 FLUTTER 框架刷新界面.

4.3.3 单词卡片生命周期的最终实现效果

因为添加单词的过程被放到后面的小节单独进行讲解了，所以在这个小节的效果展示里面只放上删除单词的效果. 单词卡片的新建动画和销毁动画在视觉效果上互为逆过程，只是动画播放完之后执行的操作不太一样而已.

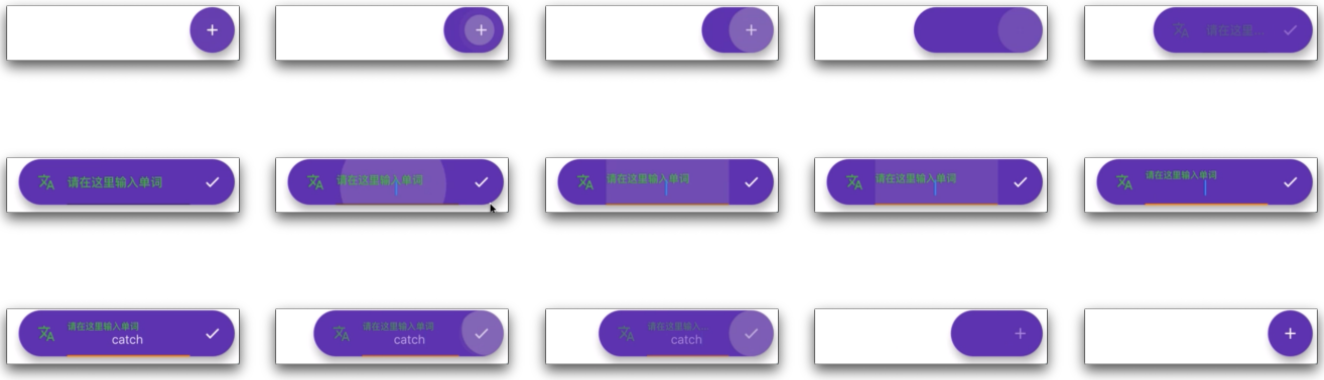


整个动画和最初设计时的想法是基本吻合的. 虽然删除卡片之后界面结构已经发生了很大的变化，但是用户却感受不到.

4.4 添加单词按钮中的细节

4.4.1 添加单词按钮最终实现的效果

下面的 15 幅图，从左至右，从上到下，按照时间顺序展示了添加单词按钮的工作方式.



4.4.2 实现上述效果的方法

✚号按钮实际上有两种状态. 两种状态的 UI 结构几乎完全不同，但是都需要被塞进同一个胶囊型容器中. 这个时候就需要良好的过渡动画来解决突兀的问题.

在点击✚号按钮时，该应用会开始播放过渡动画，这个动画只由一个动画控制器控制. 这个动画包含了形变和对于透明度的调整. 可以看到该应用在将按钮拉长的同时对按钮上的图标的透明度进行了处理，让他能够比较自然的过渡到另外一个符号. 整个动画中，该应用先调高透明度，当图标完全透明时，该应用开始调整 UI 结构，命令 FLUTTER 进行刷新，然后该应用再利用动画缓慢的将透明度调整回来.

当用户完成输入之后(输入完之后按下确定按钮)，该应用会直接将输入框中的文字发送到 WordChangeNotifier 进行一次添加单词过程(在顶层设计中已经提到过). 因为添加单词的过程是异步的，所以当哟天农户发送了添加单词的申请之后，该应用继续调整 UI 界面，将其通过一组和上面相似的过渡动画还原到没有被激活的状态.

第5章　附录

5.1　代码对应表

因为应用开发过程和编写网页很相似，中间有很多不太方便描述的细节(如果一一说明的话就太多了). 但是只像前两节那样通过抽象的文字来说明，感觉又缺了点东西(可能读者不太能感受到其中的工作量). 所以我在这里大致对每个 文件/类型 具体在做什么事进行一个大致的描述，感兴趣的读者可以结合这个表去参观一下代码，从而能够感受到每个部分更加真实的工作量.

文件	行数	作用
addWordFAB.dart	127	✚号按钮和其附带的一系列功能和动画在这个文件中实现
floatcard.dart	51	WordCard 的容器类型，实现了卡片的圆角效果和底部的阴影.
homepage.dart	79	对应了顶层设计中说明的视口(参见视口设计)
main.dart	84	APP 的入口
wordcard.dart	154	WordCard(参见单词卡片)及相关的各种动画(参见单词卡片的生命周期)实现在这里
wordChangeNotifier.dart	197	负责和数据库/网络进行交互，并在完成相应的异步工作之后通知 FLUTTER 刷新界面(参见工作机制设计)

wordfinder.dart	188	负责通过 HTTP 协议对特定网站上的信息进行爬虫(参见从网络上获取单词信息)
wordlist.dart	46	单词列表, wordChangeNotifier 在界面需要调整的时候的主要通知对象
wordpage.dart	64	单词详情页，和 WordCard 类合作完成了 Hero 动画(参见通过 HERO 动画切换单词卡片和单词详情页)

5.2 如何将我的工作和实验要求对应起来

虽然很容易发现，我的单词本实现了大量额外的功能，但是为了方便检查，这里还是列举一下我的工作时如何满足最基本的实验要求的. 额外的功能不再在这里列举.

实验要求	我的工作
可录入单词并分级	我的单词本实现了三个 wordlist，用户通过选中(选中的意思参考 Figure 9 异步机制)不同的 wordlist 再添加单词来实现对单词的分级
随机背诵所有单词	我的 wordlist 是可以随意滑动的，用户当然可以随机(物理)背诵所有单词.
随机背诵某级的单词	同上，只竖着滑动就可以只背某级的单词.
背诵分为中文背英文，英文背诵中文	我的单词本除了可以添加英文，还是可以添加中文的. 单词卡片上只会显示用户在添加单词时输入的字符串，而相关的中文/英文意思则显示在单词详情页(需要点击才能查看). 从而实现了要求的功能.

可以看到的是，我充分利用了简洁的界面，有效的完成了所有的实验要求.

5.3 如何运行我的代码

如果您想要自己运行我的代码，可能会有些麻烦，不过按照以下这些步骤，应也是可以跑起来的

1. 首先您需要通过 FLUTTER 官方给出的安装方式配置 FLUTTER 开发环境(https://flutter.dev/docs/get-started/install).
2. 安装完 FLUTTER 之后，需要安装 Android 或者 IOS 的开发环境，您可以任选 Android Studio(全平台)或者 Xcode(macOS 独占)，来生成 Android 应用或者 IOS 应用.
3. 运行 flutter doctor 检测环境是否配置正确
4. 进入我的项目文件，输入 flutter run. (如果报路径错误，可以将 pubspec.lock 删掉之后再试一遍).
5. 安装期间可能会用到 Google 服务，可能需要挂 VPN.

5.4 代码仓库

https://github.com/chenxiaoyu233/WordCard

5.5 演示视频

https://www.bilibili.com/video/av55090161