
山东大学

毕业论文(设计)

抑郁症辅助社区 APP 开发

Depression Assisted Community APP Development

姓 名 孙圣浩

学 号 201500121124

学 院 信息科学与工程学院

专 业 物联网工程

年 级 2015 级

指导教师 刘 治

2019 年 5 月 10 日

摘要

在信息时代科技与文化的发展下，心理疾病的发现率与确诊率也在稳步提高。以抑郁症为例，战胜抑郁症的重要手段之一，就是建立积极、客观看待问题的视角。为此，本设计旨在制作一个能够帮助抑郁患者获取有助康复的信息、并且为他们提供辅助测试问卷功能的 APP——Listener。其中使用了 Flutter 制作前端 APP，并用 LNMP 框架实现了后端服务。

本设计实现了：

前端：登录页面、文章浏览与阅读页面、侧边栏、聊天、收藏夹、测试页面以及界面路由与组织等；

本地数据存储：登录信息、收藏的文章；

后端：服务器购买、搭建，域名选购、SSL 配置； php 与数据库通信、sql 查询； nginx 上运载 php 脚本。

前后端通信：前端访问服务器并 get/post 报表；前后端根据所得信息，进行下一步操作。

与传统软件开发相比，本设计使用了最现代、流行的架构设计，并采用了许多开源模块，实现了快速、低成本开发与标准、现代的 UI 设计。

关键词：抑郁症，dart，flutter，LNMP，社交软件

ABSTRACT

Under the development of science and technology in the information age, the rate of discovery and diagnosis of mental illness is also steadily increasing. Taking depression as an example, one of the important means to overcome depression is to establish a positive and customer-oriented perspective. To this end, the design is designed to create an APP that can help depressed patients get information that helps them recover and provide them with an auxiliary test questionnaire. The APP, which named Listener, is made by Flutter in the front-end part, and the back-end service is implemented by the LNMP framework.

This design implements:

Front end: login page, article browsing and reading page, sidebar, chat page, favorites, test page, and interface routing and organization etc.;

local data storage: login information, favorite articles;

Backend: purchase and build cloud server machine, domain name purchase, SSL configuration; php and database communication, SQL query; run php script on Nginx server.

Front-end communication: The front-end accesses the server and gets/post reports; the front-end and the end perform the next operation based on the information obtained.

Compared with traditional software development, this design uses the most modern and popular architecture designs, and uses many open source modules to achieve fast, low-cost development and standard, modern UI design.

KEYWORDS: depression, dart, flutter, LNMP, social software

目录

第一章 绪论..... 1

1.1 课题背景 1

1.2 课题的研究现状..... 1

1.3 设计架构与软件选择 1

1.3.1 前端选择： Dart 与 Flutter..... 1

1.3.2 后端选择：LNMP..... 2

第二章 设计规划..... 4

2.1 功能设计 4

2.1.1 用户功能 4

2.1.2 资讯功能 4

2.2 架构规划 4

2.2.1 前端实现： Dart 与 Flutter 4

2.2.2 后端实现：LNMP^[10] 5

2.2.3 数据交互：http 协议与 json 串..... 5

第三章 前端实现..... 6

3.1 登录 6

3.1.1 界面布局 6

3.1.2 登录与注册 7

3.1.3 序列化与本地存储..... 8

3.2 主界面与侧边栏..... 9

3.2.1 界面设计 9

3.2.2 主显示界面 9

3.2.3 侧边栏与页面路由..... 10

3.3 文章浏览与阅读..... 11

3.3.1 主页浏览文章缩略信息 11

3.3.2 文章阅读界面 11

3.3.3 文章收藏功能 12

3.3.4 序列化与本地存储..... 13

3.4 调查与测评页面..... 13

3.5 聊天服务 13

第四章 后端实现..... 15

4.1 登录服务器 15

4.2 数据库设计 15

4.2.1 用户表.....	16
4.2.2 文章表.....	16
4.2.3 测试表.....	16
4.3 PHP 实现登录与读取.....	17
4.3.1 设计思路.....	17
4.3.2 用户注册与登录.....	18
4.3.3 读取文章与测试.....	20
4.4 Nginx 服务端.....	21
4.4.1 端口.....	22
4.4.2 SSL 证书.....	22
4.4.3 php 配置.....	22
第五章 前后端整合	24
5.1 数据交互	24
5.1.1 Json	24
5.1.2 前后端实现	24
5.2 用户登录与获取文章、测试	24
结论	25
1 成果与展示	25
2 不足与展望	25
致谢	26
参考文献	27
附录	28
1 Flutter 环境	28
1.1 获取并安装 Flutter SDK.....	28
1.2 设置 Android 开发环境	28
1.3 三方库	28
2 LNMP 环境	29
2.1 搭建 Nginx 静态服务器.....	29
2.2 安装 MySQL 数据库	29
2.3 搭建 PHP 环境.....	29

第一章 绪论

1.1 课题背景

抑郁症是种常见的精神疾患。抑郁症往往长期存在，在秋冬与阳光不足、气候阴沉的季节经常复发，会严重影响个人工作学习能力。抑郁症患者除了心情郁闷之外，还会丧失对许多曾经喜欢事物的兴趣与成就感。其尤其容易产生负罪感，自尊心不足。另外抑郁患者往往睡眠和食欲紊乱，身体精力疲倦，注意力不易集中。最严重时，抑郁症可引致自杀。轻微的抑郁症患者通过心理疏导、不需要药物即可处理，而中度或重度抑郁症患者一般则要药物治疗以及更专业的治疗。^[1]

战胜抑郁症的重要手段之一，就是与抑郁患者建立关系、推送新的信息与视角。^[4]为此，本设计旨在制作一个能够帮助抑郁患者获取有助康复的信息、并且为他们提供辅助测试问卷功能的 APP——Listener。

1.2 课题的研究现状

目前主流的开发手段有：

- 原生开发：即直接在 PC、Android、IOS 等平台上进行开发，其代码针对平台底层原理而效率高、稳定，但同时开发成本与周期较高；
- Web 开发：将 web 页面嵌入相应的软件，更加轻量级、开发高效，并且发布方便、迭代快速，缺点是性能较一般，一般用于运行轻量级应用；
- 小程序开发：基于 wechat 等平台提供的丰富的库而进行的再次开发，其运作就在开发平台的生态圈之中，优点是开发快捷、生态圈完善，弊端也很明显，就是受制于其平台；
- 跨平台开发：除了使用虚拟机实现跨平台运行的 Java 程序外，使用 JavaScript 开发的程序也可以实现跨平台运行，一些企业开源的运行库更是使开发快捷高效，如 Facebook 的 React，Google 的 Node.js、Angular，以及时下流行的 Vue 等。

1.3 设计架构与软件选择

1.3.1 前端选择： Dart 与 Flutter

1. Flutter

Flutter 是一个跨平台的移动端 UI 框架，其中使用了 Google 的 MaterialDesign 设计，旨在帮助开发者编写一套代码就能生成高性能的 Android 和 iOS 应用。^[2]

flutter 优点主要包括：

- 跨平台运行
- 开源、免费
- Hot Reload、响应式框架
- 丰富的控件以及开发工具
- 灵活的界面设计以及控件组合
- 有着高性能 ARM 代码与可以移植的 GPU 加速的渲染引擎，可实现游戏级别的高帧率与分辨率的界面显示

2. Dart

Flutter 程序使用 Dart 语言进行编写。Dart 是一门由 Google 开发并批准为 ECMA 标准的通用型编程语言。它可以用于构建 Web、服务端与移动端程序，其源代码开源、免费。

Dart 是一门面向对象的语言，它使用类和单继承，可以编译为 JavaScript。与时下更流行的 Java 相比，Dart 没有接口、抽象基类或“普通”类，Dart 只有作为接口的类。另外 Dart 允许用户自定义运算符。^[3]

Dart 主要思想之一就是可选类型，另一个关键思想就是反射。作为一门具有潜力的现代化编程语言，本设计的前端页面基于它进行部署。

1.3.2 后端选择：LNMP

1. L:Linux 作为云服务器上的操作系统，在这里选择 CentOS 发行版，来自于 Red Hat Enterprise Linux 依照开放源代码规定发布的源代码所编译而成。有些要求高度稳定性的服务器以 CentOS 替代商业版的 Red Hat Enterprise Linux 使用。也可以选择 Ubuntu，它是基于 Debian GNU 和 Linux 的免费开源桌面 PC 操作系统，支持 x86、amd64(即 x64)和 ppc 架构。它也是目前使用用户最多的 Linux 版本，用户数已超过十亿（2015，包含手机等设备）。^[5]

2. N:Nginx 是一款开源的服务器软件。Nginx 采用异步框架（意味着更好的处理并发请求），并能处理负载均衡。该软件由 IgorSysoev 创建，并于 2004 年首次公开发布，同名公司成立于 2011 年，以提供支持。相对于 Apache 来说，Nginx 占用更少的内存及资源，并且有利于高并发环境。整体设计高度模块化，编写模块相对简单、稳定，社区精品模块生产活跃。Apache 则是老牌服务器，胜任大多类型的网页需求，并且整体性能稳定、模块更多。Apache 适合处理动态请求，而 Nginx 长于处理静态请求，其对 CPU 与内存的占用率率低。

现在一般前端用 Nginx 作为反向代理抗住压力，而 Apache 作为后端处理动态请求。

3. M:MySQL 数据库管理软件，负责管理本地数据，并分配用户以读写权限。原本是一个开放源代码的关系型数据库(一种数据库规范化标准)管理系统,原开发者为瑞典的 MySQL AB 公司，2008 年被 SunMicrosystems 收购，该公司再 2009 年被甲骨文公司收购，现在 MySQL 为 Oracle 旗下的产品。它不仅能满足小型网站的需求，许多大型网站（如 wikipedia,google,facebook）也开始使用 MySQL 管理自己的数据库。（注：M 也可以是 MariaDB，它是 MySQL 的一个分支。在 MySQL 被 Oracle 公司收购后，MySQL 就有了闭源的潜在风险，因此社区采用分支的方式来避开这个风险）。由于 CentOS 从 7.0 发行版开始已将默认的数据库从 MySQL 切换到 MariaDB，所以我们可以用该数据库代替 MySQL 的 server 与 client 端。

4. P:PHP 是开源的通用计算机脚本语言，尤其适用于网络开发，常嵌入 HTML 中使用。它的语法借鉴吸收 C 语言、Java 和 Perl 等主流计算机语言的特点，易于快速上手开发。PHP 设计的初衷是帮助网络开发人员快速编写动态页面，但现在 PHP 也使用在很多其他领域。

第二章 设计规划

2.1 功能设计

本设计围绕两个核心进行：用户功能与社区功能。前者实现客户端向服务端存储、读取用户资料，后者实现文章、测评的展示。下面进行详细说明：

2.1.1 用户功能

1. 用户注册与登录

通过访问服务端相应的 url 资源，客户端可以向服务端 post 登录信息，服务端将使用 json 格式的数据进行相应的回复，包括客户信息字段、登录状态字段等。

2. 用户本地存储数据

Flutter 三方库中有一款叫做 shared_preferences 的存储管理库，可以将需要储存的信息以 Key=>Value 数据对的形式存在本地。通过调用该库，并进行本地化设计，可以将用户相关信息存在本地，再次打开软件时进行读取，便可以实现自动填写登录字段。

2.1.2 资讯功能

1. 服务端资讯访问页面

资讯信息保存在服务器的数据库中，可以使用 php 脚本调用数据库，并使用 json 格式的串与客户端进行数据交互。

2. 客户端展示页面

客户端在获取来自服务端的数据后，还需要进行 decode、页面排版与展示。包括浏览界面、详细阅读界面、收藏夹页面等。客户还可以对文章进行相应操作，如收藏、保存、上传等。

2.2 架构规划

2.2.1 前端实现：Dart 与 Flutter

- IDE: Android Studio 3.4
- SDK: Flutter 1.2.2

2.2.2 后端实现

后端运行在使用 Linux 系统的云服务器上，采用 LNMP 架构^[10]：

- 域名：cvm.cakipaul.com（CVM: Cloud Virtual Machine）
- 服务器：腾讯云服务器
- 操作系统：CentOS7.5
- 服务端：Nginx1.12.2
- 数据库：MySQL8.0
- 后端脚本处理：PHP5.6

2.2.3 数据交互：http 协议与 json 串

作为前后端数据交互最常用的手段之一，http 协议常用于客户端与服务器之间的数据交通。客户端可以使用 get、post 等方法与服务端进行数据交互。

服务器在得到来自客户端的数据报表后，会根据其中的字段信息进行相应查询、处理操作，并向客户端返回 json 格式的返回数据。客户端在接收数据后，通过 json_decode，就可以生成 Map 键值对形式的数据，从中读取关键字所对应的信息内容^[5]。

第三章 前端实现

3.1 登录

3.1.1 界面布局

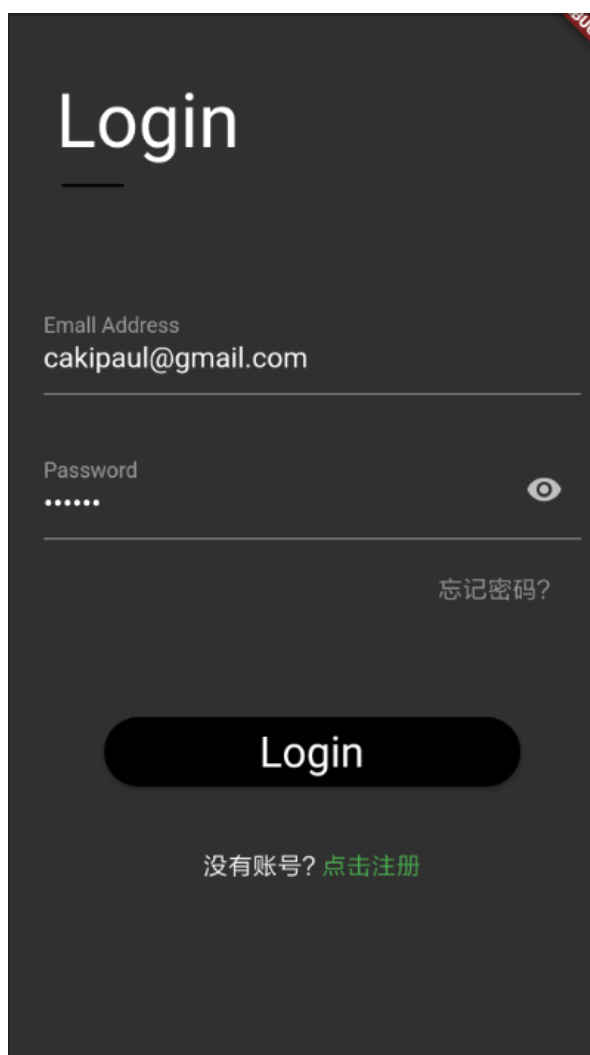


图 3.1：登录界面布局

其中包括：

- Login 的 logo
- 邮箱文本区
- 密码文本区
- 登录按钮
- 注册按钮

邮箱文本区具备文本格式检测功能，如果输入文本不符合邮箱格式将会提示格式错误，

如图 3.2。同时密码区也需要满足非空。密码文本区右侧添加眼睛 icon，点击可以切换明文，如图 3.3。

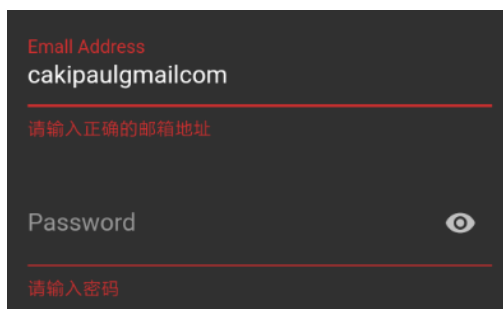


图 3.2: 输入文本格式错误示例图

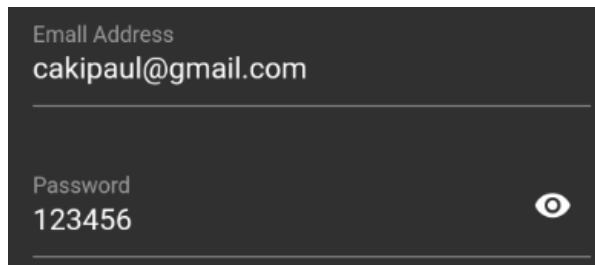


图 3.3: 明文输入密码

3.1.2 登录与注册

1. 登录页面

点击 login 按钮后，软件便会根据文本框中的信息向服务器 post 表单，然后就可以得到服务器返回的信息。根据返回的信息，用户将得到密码错误等包含可能的错误信息的 dialog，或者跳转到主界面。如图 3.4，图 3.5。

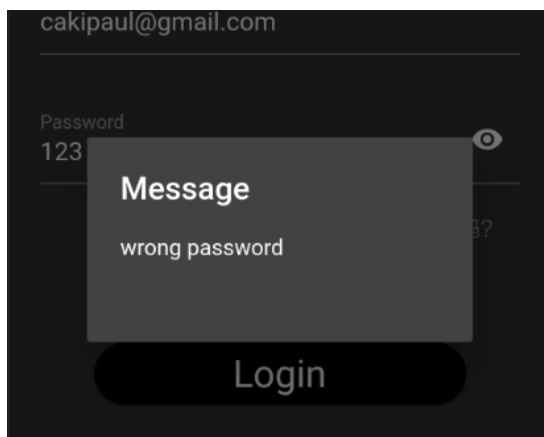


图 3.4: 输入密码错误示例图

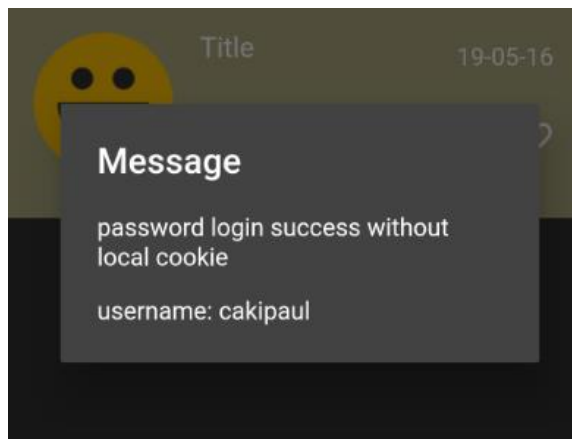


图 3.5: 密码正确，跳转到主界面

如果返回信息中包含 username 字段，则在 message 的下一行显示“‘username:’+\$username”格式的字符串；否则，显示为空行。如在图 3.4 中，message 为 wrong password 时，返回的 username 字段为 null，第二行则未显示 username 信息。

2. 注册页面

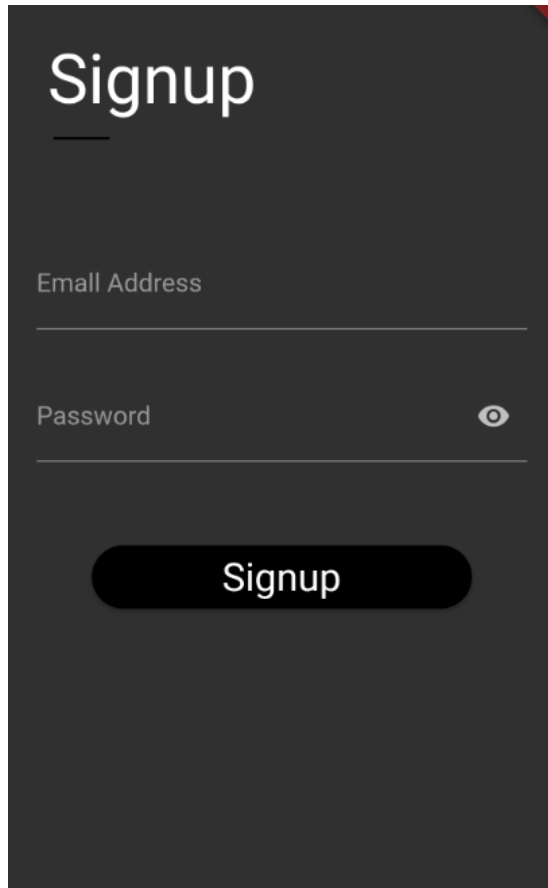


图 3.6: 注册界面布局

与登录页面类似，按照格式填写好文本字段后点击 `signup`，便会向服务器发送报表，若收到包含 `success=1` 的返回字段，则说明注册成功，需要跳转到自己邮箱进行下一步验证。

3.1.3 序列化与本地存储

在点击 `login` 按钮后，将把文本框中的信息以键值对的形式保存在本地。在初始化页面时，查询本地存储，并自动填充文本框。Debug 存储示例如图 3.7。在初始化登录界面时，查询本地 `email` 关键字所指内容，并填写到文本框中；点击 `login` 后，将更新后的 `email` 信息存在本地。

```
Installing build\app\outputs\apk\app.apk...
Syncing files to device Redmi Note 5...
I/flutter ( 6137): 本地email读取成功: cakipaul@gmail.com
I/flutter ( 6137): email本地存储成功: TextEditingController#82bfd(TextEditingValue(text: | cakipaul@gmail.com |
```

图 3.7: 本地 email 存储与读取

3.2 主界面与侧边栏

3.2.1 界面设计

其界面布局如图 3.8、图 3.9。

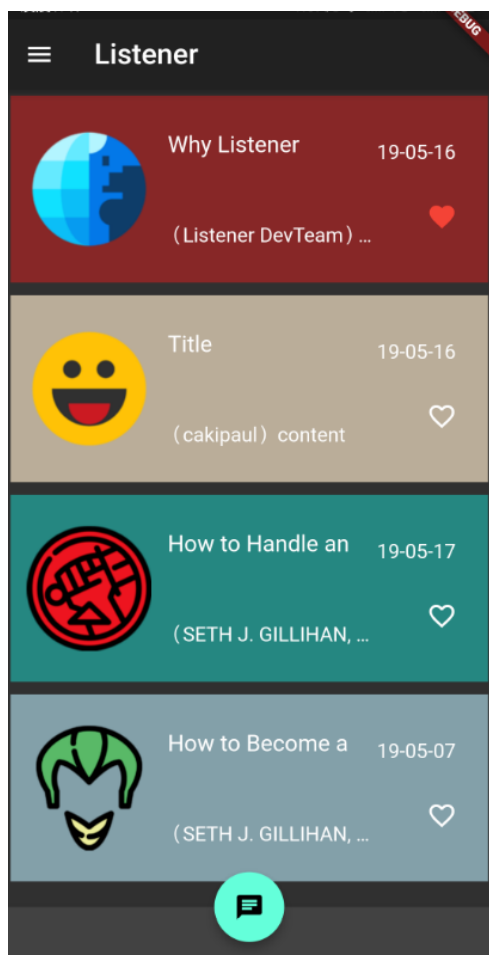


图 3.8：主显示界面

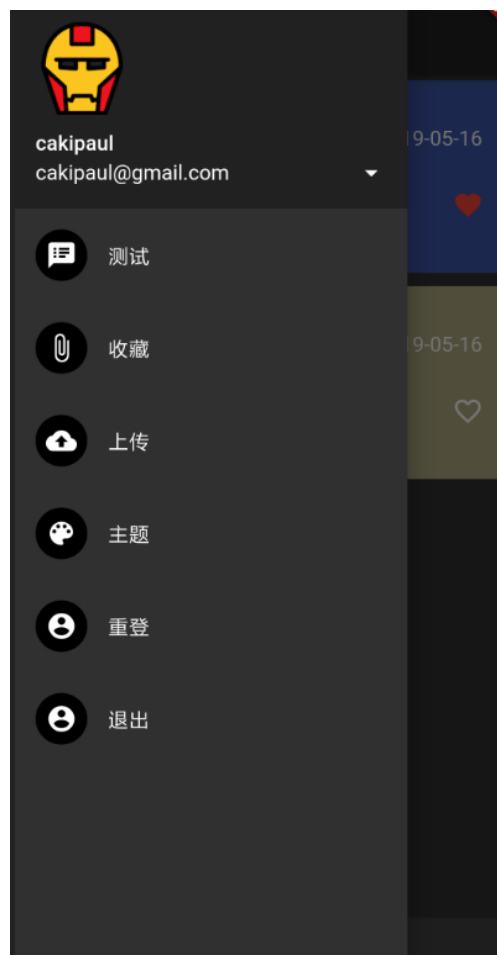


图 3.9：侧边栏

3.2.2 主显示界面

主显示界面（图 3.8）展示文章列表，下方有一个浮动按钮，可以链接聊天服务器；左上角有菜单键，点击打开侧边栏。

其中文章列表每个条目的背景颜色随机变动，色彩采用饱和度较高、亮度较低、具有一定卡通质感的颜色。

3.2.3 侧边栏与页面路由

1. 侧边栏

侧边栏包含以下项目（图 3.9）：

- 用户头像、用户名与邮箱
- 文章收藏夹
- 测试入口
- 上传信息
- 主题设置
- 注销重登
- 退出软件
-

其中用户相关的信息在登录时获取。

主题状态的刷新使用 flutter-provide 库进行管理。它是谷歌出品的一个状态管理框架，它允许在小部件树中传递数据。在点击“主题”按钮时更新主题对象内容，并刷新主界面的 state。

其他部分则通过页面路由，导向新的页面。

2. 页面路由

页面之间的跳转通过路由实现。在 Flutter 的主 App 中可以设置 Route。Flutter 的路由以树状结构展开，其命名类似文件绝对路径，由斜线“/”进行分级，并以之开头。^[14]本设计的文件路径规划如图 3.8.

```
routes: <String, WidgetBuilder>{
  '/signuppage': (BuildContext context) => new SignupPage(),
  '/loginpage': (BuildContext context) => new LoginPage(),
  '/main': (BuildContext context) => new MainLayout(),
  '/testpage': (BuildContext context) => new Tests(),
  '/chatpage': (BuildContext context) => new ChatServer(),
  '/likedpage': (BuildContext context) => new LikedArticles(),
},
initialRoute: '/loginpage',
```

图 3.10: 本地 email 存储与读取

3.3 文章浏览与阅读

3.3.1 主页浏览文章缩略信息

展示文章（图 3.8）列表展示文章的简略信息与收藏情况，包括：

- 展示图片：pic
- 标题：title
- 作者：author
- 写作日期：date
- 是否被收藏：icon.heart

每一篇文章的信息包含在卡片条目中，条目之间有着清楚的分界线。可以通过下滑进行刷新，浏览更多文章。

3.3.2 文章阅读界面

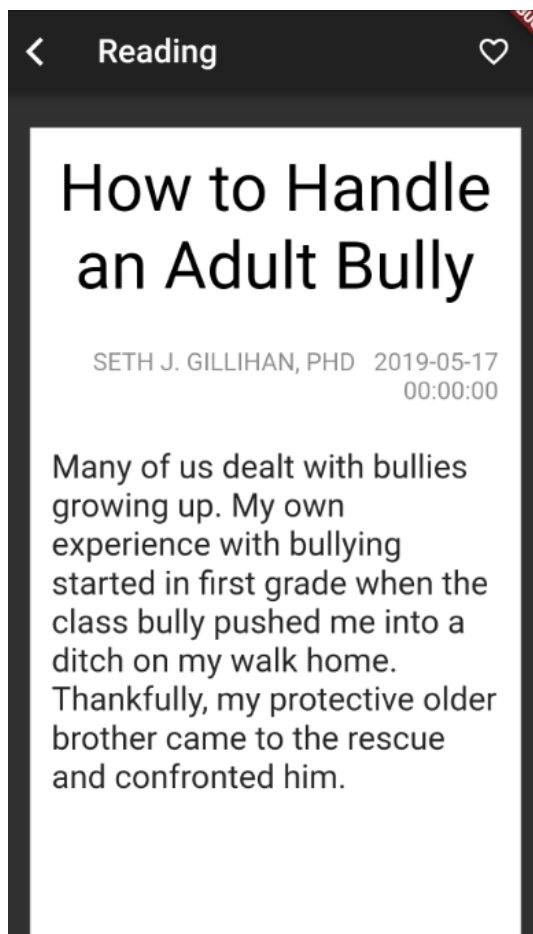


图 3.11：文章阅读界面

在主页或收藏夹浏览文章时，点击文章后将跳转到文章详细阅读页面。其中包含了文章标题、作者、日期以及正文信息。点击右上角的心形可以将文章添加到收藏夹。

3.3.3 文章收藏功能

在文章阅读界面中，点击右上角的心形便可以将文章添加、移除收藏夹。已添加到收藏夹的文章可以看到红色的心形，并能在收藏列表中查看，如图 3.12。

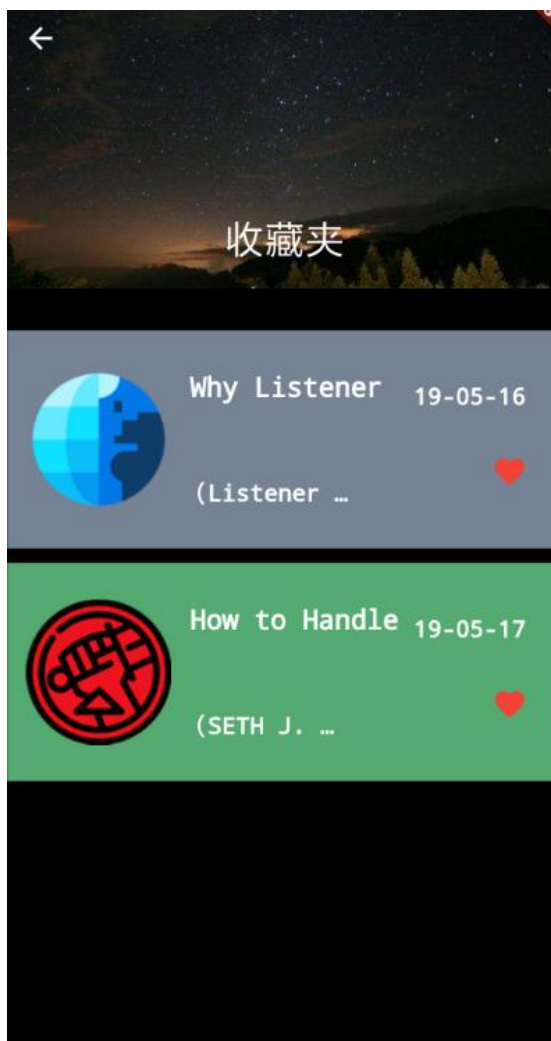


图 3.12: 文章收藏界面

为了防止文章重复收藏，需要使用 Set 集合进行管理（而非 List）。另外为了实现 Set 集合元素不重复的功能，需要重写储存文章内容的 Article 类中的 hashCode() 与 == 运算符方法。

收藏与移除的 debug 示例如图 3.13。

```
I/flutter (14628): 收藏成功
I/flutter (14628): fav: {Article}
I/flutter (14628): 收藏成功
I/flutter (14628): fav: {Article, Article}
```

图 3.13: 添加收藏 debug 调试展示

3.3.4 序列化与本地存储

为了实现本地收藏夹，需要在添加、移除收藏夹时更新本地存储，并在 login 操作时读取本地存储。其 debug 示例如图 3.14、图 3.15。

```
I/flutter (15571): 移除成功
I/flutter (15571): fav: {Article}
I/flutter (15571): 正在存储第 1个, 序号: 10000
I/flutter (15571): 本地存储成功: {article_id:10000: {title: Title, author: cakipaul, content: content, pic: h
I/flutter (15571): 收藏成功
I/flutter (15571): fav: {Article, Article}
I/flutter (15571): 正在存储第 1个, 序号: 10000
I/flutter (15571): 正在存储第 2个, 序号: 9999
I/flutter (15571): 本地存储成功: {article_id:10000: {title: Title, author: cakipaul, content: content, pic: h
```

图 3.14: 储存本地收藏 debug 调试展示

```
I/flutter (15571): 读取本地收藏: {article_id:10000: {title: Why Listener, author: Listener DevTeam,
```

图 3.15: 读取本地收藏 debug 调试展示

3.4 调查与测评页面

为了协助咨询，需要用户做一些相应的调查问卷。问卷的形式可以是选择，也可以是简答等，所以调查与测评页面的设计就比较灵活。一种是直接采用 web 页面的方法，将调查问卷内容以网页形式展示；另一种就是使用 markdown 渲染，以纯文本的方式组织问卷。其页面与上面的文章界面类似，不再赘述。

3.5 聊天服务

主界面(图 3.8)下方的按钮通向聊天服务器，目前实现了与 ws://echo.websocket.org 网站的通信，可以显示该网站回复的内容，而其内容正是所发送文本的复制。未来可以在此基础上进行拓展，将导向的网页 url 修改为后端聊天服务资源网址，即可实现简单的在线聊天功能。

第四章 后端实现

4.1 登录服务器

我们可以通过 Windows 下的 CMD 命令控制行（通过 win+r，输入 cmd 打开，一般在搜索栏或根目录下通过管理员身份打开）来与服务器建立链接，输入以下代码：

```
ssh root@你的主机 ip 或域名 -p 22
```

其中 SSH 是 Secure Shell（安全外壳协议，简称 SSH）的缩写，它采用加密的网络传输协议，可在不安全的网络中为网络服务提供安全的传输环境。任何网络服务都可以通过 SSH 实现安全传输，但 SSH 最常见的用途是远程登录系统。

root@ip -p 22 该字段表示“IP 地址（ip）所指向服务器的 22 端口（-p 22）上所运行程序的用户（root）”，即唯一标识了自己云主机上 CentOS 系统运行的 SSH 服务与 root 用户。

由于 SSH 服务默认监听 22 端口，若没有手动修改过，-p 22 字段便可以省略。另外本地 hosts 文件中可以添加云服务器 ip cvmHost 项目，下一次使用 SSH 登陆便可以直接采用如下语句：

```
ssh root@cvmHost
```

另外对 Linux 主机来说，有许多免费强大的软件可以配合建立与服务器的连接，其中比较突出的有 Xshell 与 Xftp，截止到 2018.12 该软件已更新到第六版。前者提供功能强大的命令行工具，后者提供文件传输的图形窗口。下载好软件之后，输入云主机 ip、端口（ssh 默认 22，ftp 默认 20/21）、用户名（root）、密码等就可以与云服务器建立连接了。

4.2 数据库设计

使用数据库可以实现前后端的数据分离。数据库中的数据存储在“表（table）”中，存储数据的过程需要数据库服务器提供服务。一个数据库服务器可以管理多个数据库，开发人员一般会为每个应用创建一个数据库，并在每个数据库中创建多个表。^[6]本设计需要在数据库服务器上建立“listener”数据库。

使用数据库的好处有：

- 1 数据结构化：最重要的特征。数据不是面向某个应用，而是面向全组织；
- 2 数据共享：减少数据冗余，避免数据不一致；
- 3 数据独立：分为逻辑独立与物理独立两点；
- 4 数据统一管控：安全控制、完整控制、并发控制。

本设计需要用到三个表，分别存储用户数据、文章数据、测试数据，每个表中的字段参数如下文所述。

4.2.1 用户表

用户表命名为 users。字段信息见图 4.1。

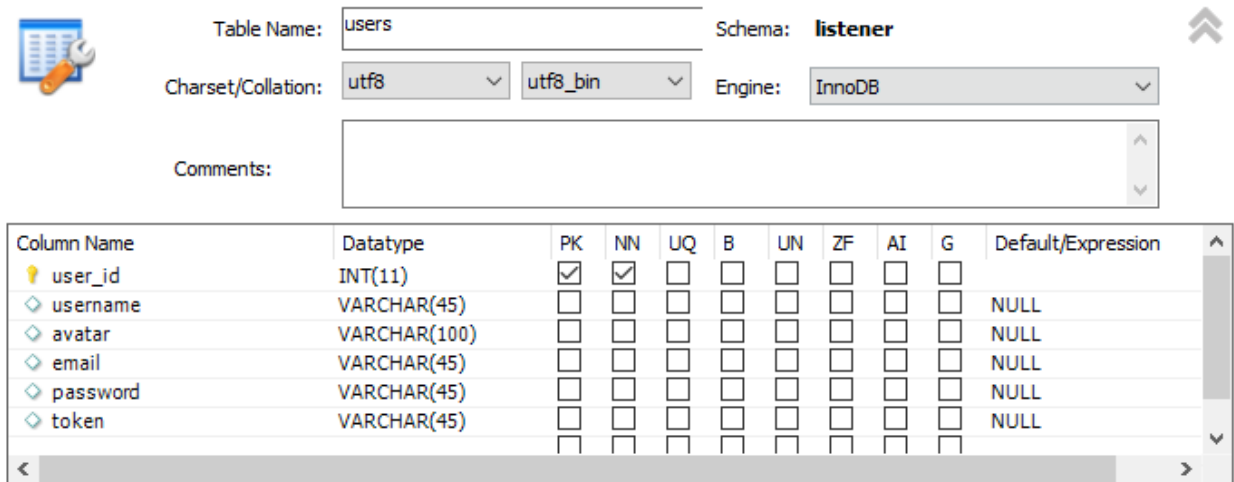


Table Name: Schema: **listener**

Charset/Collation: Engine:

Comments:







Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 user_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 username	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 avatar	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 email	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 password	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 token	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

图 4.1：用户表

4.2.2 文章表

文章表命名为 articles。字段信息见图 4.2。

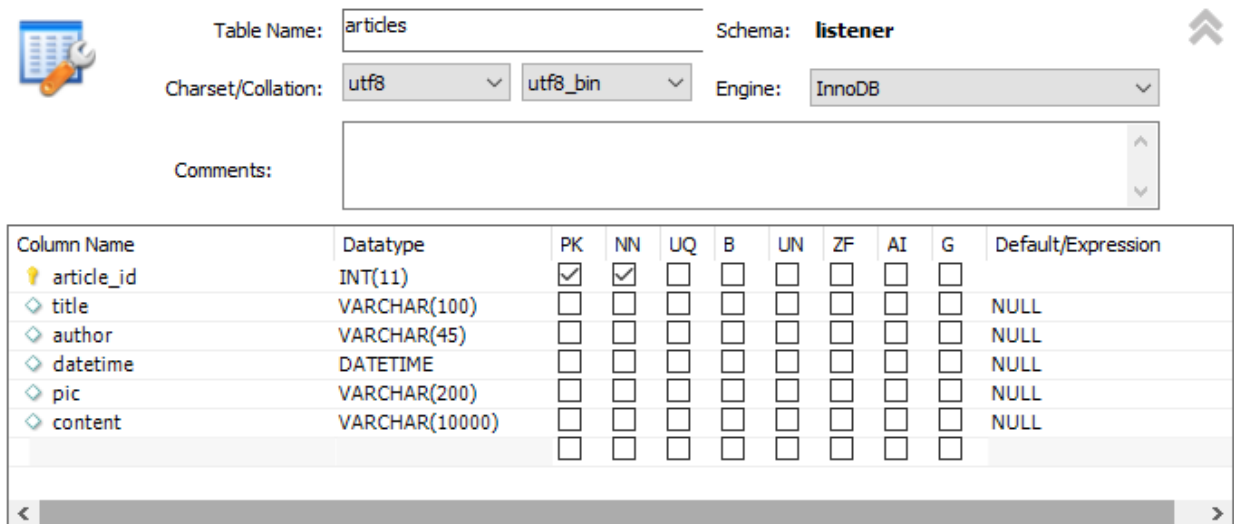


Table Name: Schema: **listener**

Charset/Collation: Engine:

Comments:







Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 article_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 title	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 author	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 datetime	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 pic	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 content	VARCHAR(10000)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

图 4.2：文章表

4.2.3 测试表

测试表命名为 tests。字段信息见图 4.3。

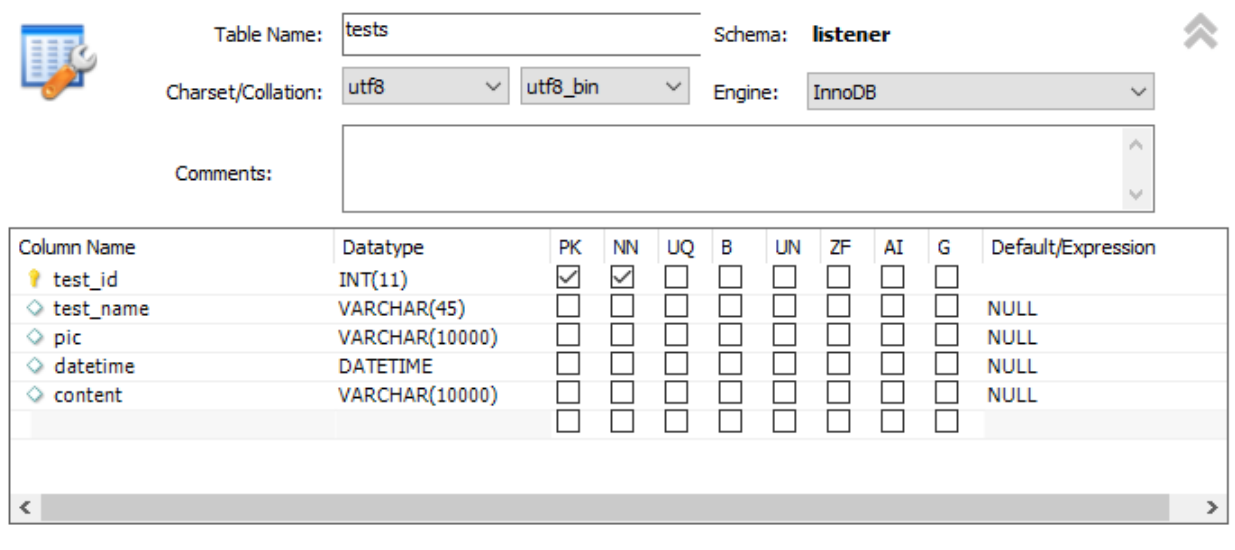


Table Name: Schema: **listener**

Charset/Collation: Engine:

Comments:






Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 test_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 test_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 pic	VARCHAR(10000)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 datetime	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 content	VARCHAR(10000)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

图 4.3：测试表

4.3 PHP 实现登录与读取

4.3.1 设计思路

我们将登录流程进行分解，然后把每个环节所需要实现的功能分装在各个脚本中。其流程如下：

1. 登录流程：condb.php->check.php->login.php
2. 获取文章：condb.php->getArticles.php
3. 获取测试：condb.php->getTests.php

其流程如图 4.4。

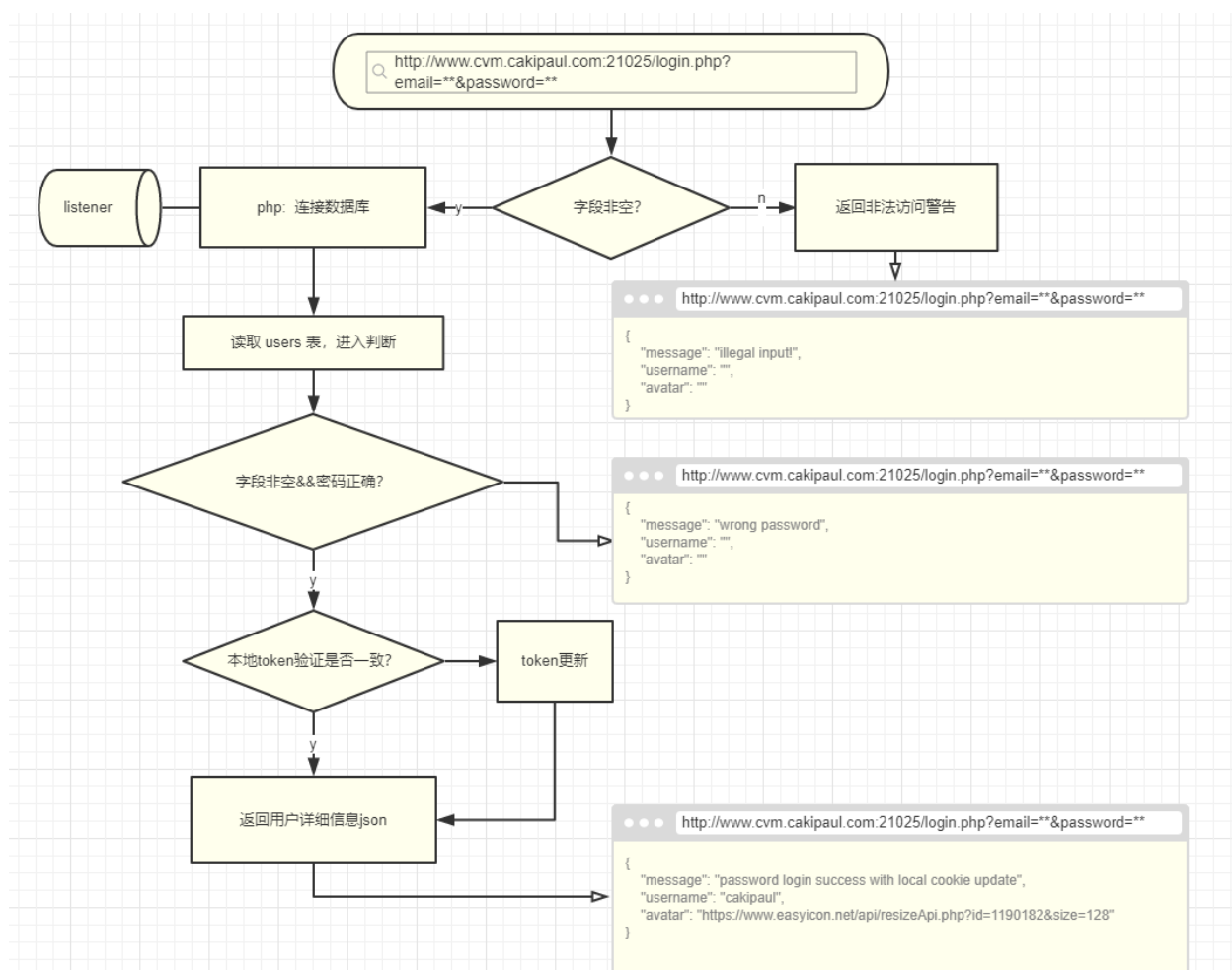


图 4.4: 后端登录流程示意

其中 check.php 中包装了许多常用 function，如通过 email 查询 username、检查密码是否正确等。getArticles.php 返回包含文章信息的 json 串，getTests.php 则返回包含测试内容的 json 串。

将连接数据库所需要的信息封装在 condb.php 文件中，其中包含以下参数；

- Servername
- Username
- Password
- Dbname
- program_char = "utf8"

依据这些信息，就可以使用 php 新建一个 mysqli 数据库连接。然后依据前端的 get/post 请求进行相应的查询与验证操作。

4.3.2 用户注册与登录

用户登录需要验证前端 post 来的表单字段的正确性。后端使用连接数据库的 php 脚本

后，会得到一个数据库连接对象，对它进行相应字段的查询操作，对比客户端发来的用户信息，就可以判断其 email 与 password 是否正确。如果客户端发送的两字段有一个为空，则返回非法登录警告；否则，根据密码是否正确，返回相应的 message 字段。另外在密码正确时还返回用户的详细信息。

除了使用用户 email 与 password 登录，还可以在数据库中加入 token 字段，用于本地免密登录。在首次登陆后，本地会存储该字段，如果本地再次发起服务请求，服务端会检查本地的 token 字段与服务器端是否相同，如果相同，则表示登陆成功。该字段会在再次登陆后进行刷新，并且会根据设置的间隔进行定期刷新。

将实际登录情况进行分类如下：

1. 首次登录：用户名+密码
2. 再次登录：token 验证登录
3. 异地登录：用户名+密码，token 更新
4. 长时间未登录：用户名+密码，token 更新

新用户注册流程与登录流程类似，只不过在向数据库添加新用户信息前，需要通过邮箱服务进行邮箱验证。服务器将向注册邮箱发送注册确认链接，用户需要在指定时间（如 24 小时）内点击该链接进行邮箱激活，然后服务器才会将该用户信息添加到数据库中；否则注册失败。

使用 PostMan 来进行后端调试。第一次登陆时会保存本地 cookie，返回 json 串的 message 字段为“password login success with local cookie update”如图 4.5。

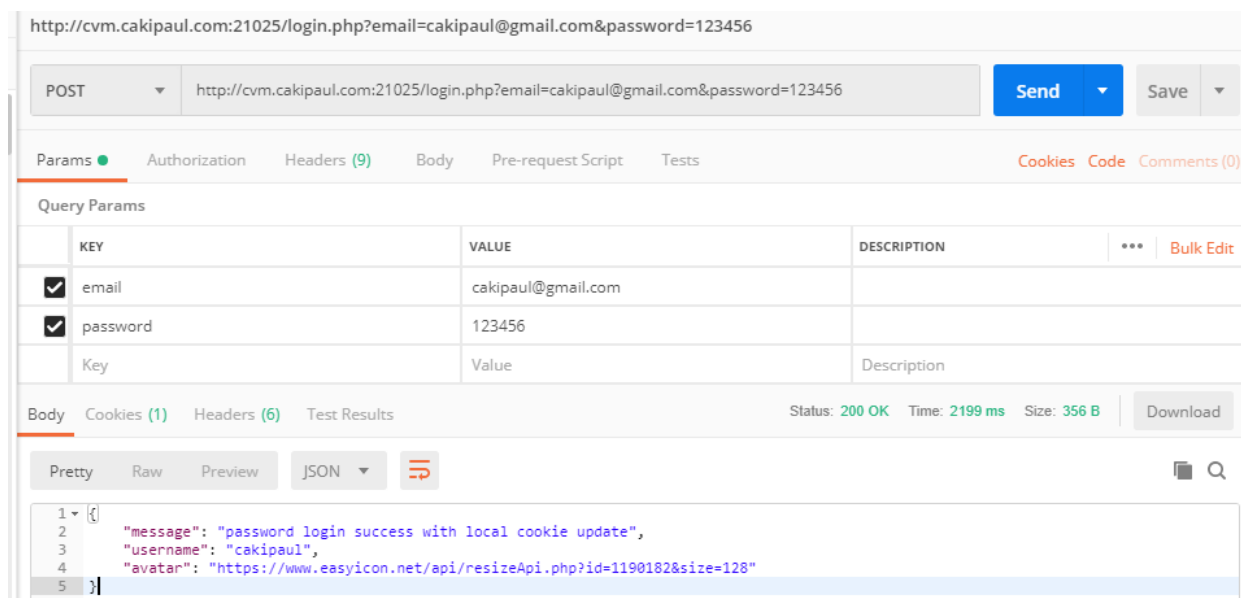


图 4.5：使用密码登录，本地 cookie 更新

再次登录时，message 字段为“token login success”，表示使用本地 token 登陆成功。如图 4.6。

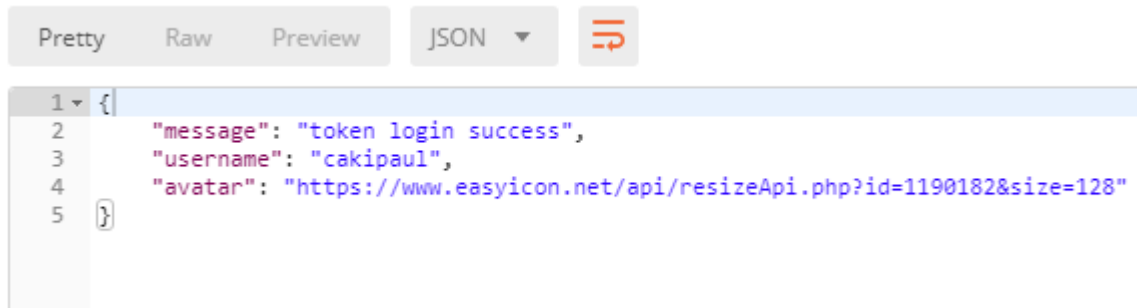


图 4.6: 使用本地 token 登录成功

Body Cookies (1) Headers (6) Test Results							Status: 200 OK	Time: 1315 ms	Size: 328 B	Download
Name	Value	Domain	Path	Expires	HttpOnly	Secure				
token	a006f0f858d3950826249b33cc51a388		/	Sat, 18 May 2019 00:06:48 GMT	false	false				

图 4.7: 本地存储 cookie 中的 token

查看服务器数据库 users 表中的项目，可以看到本地 token 与服务器数据库中相应 users.email 字段的 token 的确一致（图 4.6，图 4.7）：

```

> SELECT * FROM listener.users

***** 1. row *****
user_id: 1
username: cakipaul
avatar: https://www.easyicon.net/api/resizeApi.php?id=1190182&size=128
email: cakipaul@gmail.com
password: 123456
token: a006f0f858d3950826249b33cc51a388
    
```

图 4.5: 服务器数据库中用户信息

删除 cookie 后，再次登录则是使用邮箱与密码登录。

4.3.3 读取文章与测试

与用户登录类似，服务器上的文章通过 php 脚本，从数据库的 articles 表中进行读取。其中各字段包含文章的全部信息。

访问 `cvm.cakipaul.com:21025/getArticles.php` 即可得到文章 json 串，获取文章成功时，json 串中的 success 字段值为 1。

每篇文章的具体信息包含在内容为 `”article_id:’+$id”` 的 Key 所对应的 Value 中。其中 \$id 从 10000 开始降序排列。获取文章示例如图 4.8。

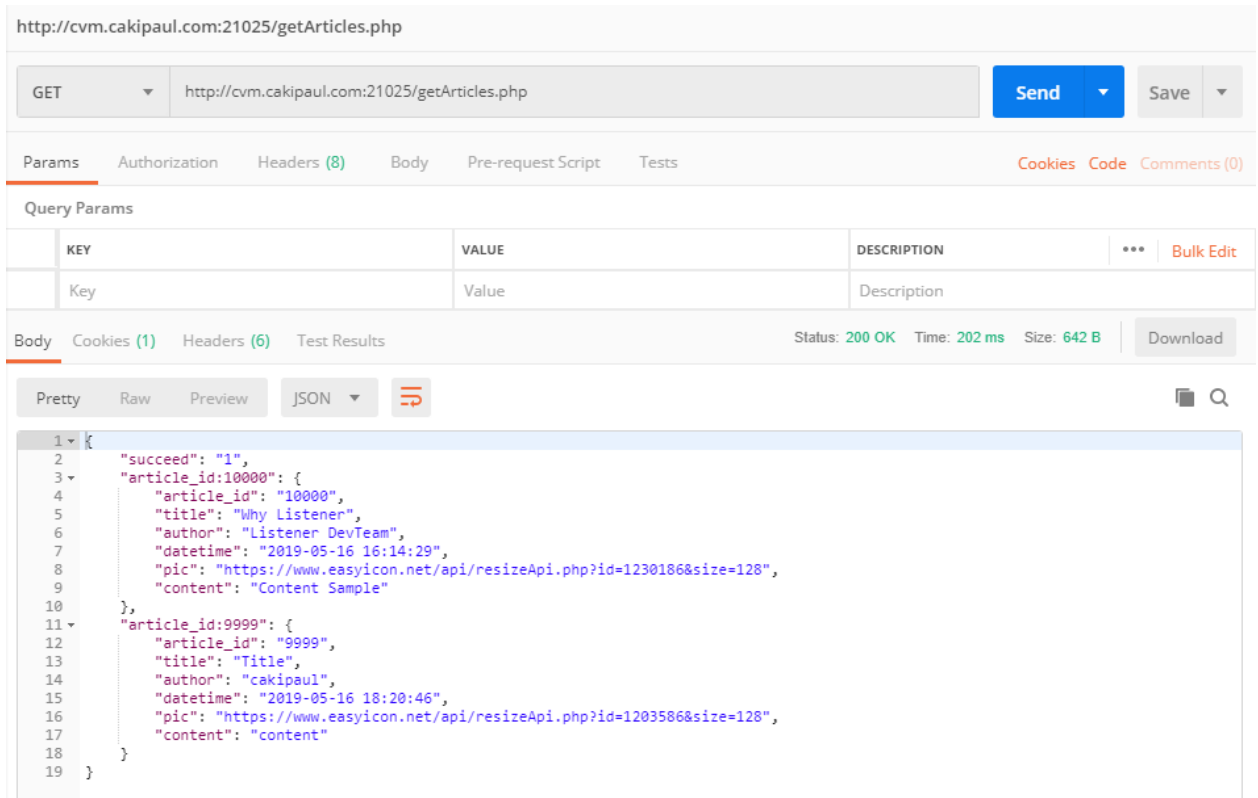


图 4.8: 调用 getArticles.php 文件获取文章

4.4 Nginx 服务端

服务器是运行在网络主机（Network Host）上的计算机软件，可以管理资源并为用户提供服务。通常分为文件服务器（能使用户在其它计算机访问文件），数据库服务器和应用程序服务器。

对网页服务器来说，每一个网页服务器程序监听 80 端口，从网络接受 HTTP 请求，然后提供 HTTP 回复给请求者。HTTP 回复一般包含一个 HTML 文件（HyperText Markup Language，是一种标记语言，可以形成标准网页）的同时也可以包含纯文本文件、图像或其他类型的文件。一般来说这些文件都存储在网页服务器的本地文件系统里，当正确安装和设置好网页服务器软件后，服务器管理员需要指定一个本地路径名为网页文件的根目录。网页服务器便会在此目录中组织、查找、运行、编辑文件。

2018.12 市场服务器占有率：（来自 w3techs 的数据）

Apache（44.6%）Nginx（40.7%）Microsoft-IIS（9.0%）LiteSpeed（3.7%）GoogleServer（0.9%）

这五种服务器在市场总占有率为 98.9%，剩余服务器约为 1.1%。其中 Apache 与 IIS 近几个月有小幅降低，Nginx 有小幅增长。本设计正是采用 Nginx 服务器。

4.4.1 端口

通过配置 Nginx 的配置文件，重启 Nginx 服务，我们可以使它在指定端口上监听 http 连接请求，并相应地在指定的文件目录上。本设计采用 21025 端口，为了纪念 StarBound 的 IP 直连默认端口。

4.4.2 SSL 证书

在 Trust Asia TLS RSA CA 网站上可以获得时效一年的免费 SSL 证书，这样就可以将站点从 http 升级到 https 链接了。

使用 KeyManager 查看证书列表如图 4.9。

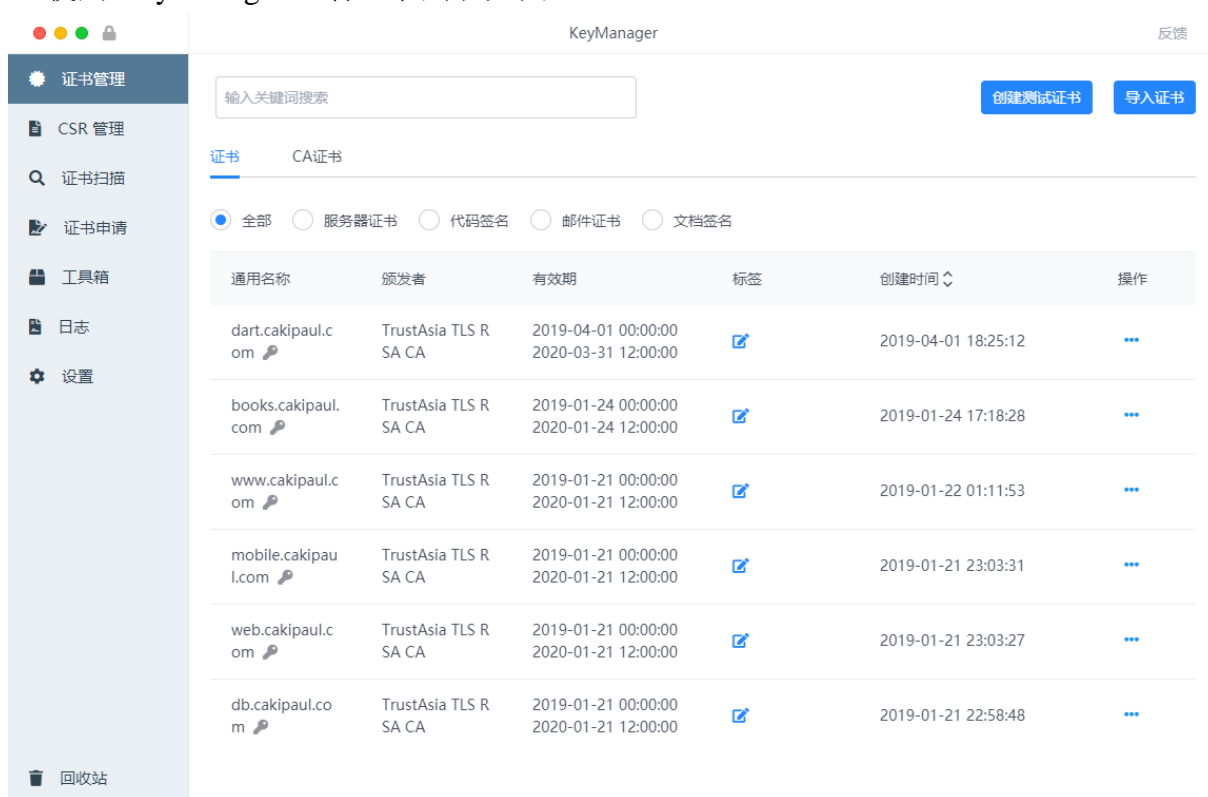


图 4.9: SSL 证书列表

4.4.3 php 配置

要想在 Nginx 上访问运行 php 脚本，需要安装 php-fast cgi 软件，并配置 Nginx 的相关文件。CGI（Common Gateway Interface，通用网关接口）接口可以让一个客户端，通过 get 与 post 等 http 请求向网络服务器上的程序进行数据交互。

从 Web 服务器的角度看，CGI 是在特定的 url 上定义了可以运行脚本程序。当收到一个匹配 url 的请求时，url 所指向的程序就会以从客户端收到的数据作为输入被执行。Web 服务器会收集程序的输出，并加上合适的 header 发送回客户端。

CGI 协议解决了 php 解释器与 webserver 通信的问题。在这之后又出现了 CGI 的改良版本：fast-cgi，也是现在最常用的处理动态网页的技术。php-fpm 是 php-Fastcgi Process Manager 的缩写，它是 FastCGI 的实现，提供了 master 进程和 worker 进程这两种进程的管理功能。master 进程负责监听端口，且只有一个，用于接收来自 Web Server 的请求；而 worker 进程则可以有多多个。

通过配置 nginx 的 nginx.conf 文件，我们可以将*.php 文件交给 php-fpm 来处理。

第五章 前后端整合

5.1 数据交互

5.1.1 Json

Json 采用了易于让人阅读的文字为基础，可以用来传输由属性值或者序列性的值组成的数据对象。其内容采取键值对的形式表示，格式简单，常用于前后端之间的通信与序列化保存。

5.1.2 前后端实现

在前端程序中，使用 `dart:convert` 就可以手动序列化 JSON。 `dart:convert` 库中包含一个简单的 JSON 编码器和解码器。

后端 php 中可以直接使用 json 的 `encode` 与 `decode` 方法，来进行 json 串与 map 集合之间的数据格式转换。

5.2 用户登录与获取文章、测试

Flutter 有个非常出名、广受好评的第三方库，叫做 Dio。它是 Flutter 中文网开源的一个强大的 Dart Http 请求库，支持 Restful API、FormData、拦截器、请求取消、Cookie 管理、文件上传/下载、超时等。这里就使用了该库中关于 http 链接的 `get`、`post` 等方法。用户数据获取测试如图 5.1。

```
I/flutter ( 7668): 本地email读取成功: cakipaul@gmail.com
I/flutter ( 7668): 本地password读取成功: 123456
I/flutter ( 7668): 本地存储并登录: email:cakipaul@gmail.com password:123456
I/flutter ( 7668): url: http://132.232.213.145:21025/login.php?email=cakipaul@gmail.com&password=123456
I/flutter ( 7668): data: {"message": "password login success without local cookie", "username": "cakipaul"}
I/flutter ( 7668): headers: connection: keep-alive
I/flutter ( 7668): x-powered-by: PHP/5.6.39
I/flutter ( 7668): set-cookie: token=735636c851b0c49e5ad0b5813a4ecfca; expires=Sat, 18-May-2019 00:06:57 GMT; Max-Age=86400; path=/
I/flutter ( 7668): date: Fri, 17 May 2019 00:06:57 GMT
I/flutter ( 7668): transfer-encoding: chunked
I/flutter ( 7668): content-type: text/html; charset=utf-8
I/flutter ( 7668): server: nginx/1.12.2
I/flutter ( 7668): request: Instance of 'RequestOptions'
I/flutter ( 7668): after clear: {}
I/flutter ( 7668): message: password login success without local cookie    username: cakipaul
```

图 5.1：客户端获取用户信息

结论

1 成果与展示

本设计基本实现的预期的功能，并在此基础上使用了精美的 UI 界面。其中实现前后端各自的内容与它们之间的通信都需要不少的工作，本设计基本实现了：

1. 前端

- a) 界面：登录页面、文章浏览与阅读页面、侧边栏、聊天、收藏夹、测试页面等；
- b) 本地数据存储：登录信息、收藏的文章；
- c) 界面路由与组织；

2. 后端

- a) 服务器购买、搭建，域名选购、SSL 配置；
- b) php 与数据库通信、sql 查询；
- c) nginx 上运载 php 脚本；

3. 前后端通信

- a) 前端访问服务器并 get/post 报表；
- b) 前后端根据所得信息，进行下一步操作。

2 不足与展望

首先是注册与找回密码功能没有深入开发，新的用户需要手动添加到服务器的数据库中。之所以这样，是因为注册需要进行用户的邮箱验证。虽然有 MailGun、SendGrid 等三方免费邮件发送提供商，但是填写表单、发送邮件的功能目前尚未有空余时间实现。

另一个就是测试页面比较简单，还有丰富的提升空间，比如更细致的选项系统设计、动态特效等。

最后就是云备份功能尚未实现，受限于服务器的性能与本设计的时间，目前尚未完善。但原理上与用户注册与登录基本相同。

致谢

首先感谢母校一直以来努力建立的优美的学习环境，以及导师刘治老师对自己的指导与帮助。还要感谢舍友们营造的轻松的生活氛围。

另外生活在信息时代，感谢开源开发者们创造的宝贵财富，仅仅本设计就涉及了：

- Dart, Flutter, Linux, MySQL, php, Nginx
- 来自于 easyicon.net 的缩略图
- 网络图片提供方（baidu.com）
- Flutter 丰富的三方库

开源者们行为的意义与我的感谢之情便不再多谈，在这里一并致以感谢。

参考文献

- [1] 世界卫生组织. https://www.who.int/mental_health/management/depression/zh/.
- [2] 亢少军. Flutter 技术入门与实战[M]. 北京：机械工业出版社，2019.
- [3] Gilad Bracha. Dart 编程语言[M]. 北京：电子工业出版社，中国工信出版社，2019.
- [4] 王国荣. 与抑郁症握手言和[M]. 武汉：武汉大学出版社，2015.
- [5] 鸟哥. 鸟哥的 Linux 私房菜——服务器架设篇（第三版）[M]. 北京：机械工业出版社，2018.
- [6] 传智播客高教产品研发部. MySQL 数据库入门[M]. 北京：清华大学出版社，2015.
- [7] 程子清. Flutter 开发智慧城市相关 APP 具体实践[J]. 电子技术与软件工程, 2018(19):56-57.
- [8] 王东. 浅谈 LNMP 架构在校园网站建设中的应用[J]. 电子世界, 2018(08):151.
- [9] 孙明喆, 马国强. 基于 LNMP 的企业私有云系统[J]. 信息技术与信息化, 2017(Z1):83-84.
- [10] 刘宝莲. 基于 LNMP 搭建 Discuz 技术分享论坛[J]. 电脑知识与技术, 2018, 14(34):30-31.
- [11] 陈镭. 基于 LNMPA 架构的 Web 系统设计与实现[J]. 计算机时代, 2014(08):22-23+26.
- [12] Walrath, Kathy, Ladd, Seth. What is Dart?[J]. O'Reilly Media, 2012(1): 20.
- [13] Buckett, Chris. Dart in Action[J]. Manning Publications, 2012(1): 475.
- [14] Flutter Docs. <https://flutter.dev/docs/development/>.

附录

1 Flutter 环境

1.1 获取并安装 Flutter SDK

1. 在 github 上下载最新版本的 Flutter SDK: <https://github.com/flutter/flutter> ;
3. 解压 zip 文件, 执行 flutter_console.bat 批处理脚本;
3. 更新 path 环境变量: 将 flutter\bin 所在的完整路径添加到系统环境变量中。
4. 运行 flutter doctor

1.2 设置 Android 开发环境

1. 下载并安装 Android Studio;
2. 配置 Android 设备:
 - 2.1 在设备上打开 **Developer options** 和 **USB debugging** 选项;
 - 2.2 在命令行中, 使用 flutter devices 命令来查看 Flutter 能够识别出的设备;
3. 配置 Android 模拟器: 启动 **Android Studio > Tools > Android > AVD Manager**, 然后选择 **Create Virtual Device** 选项, 便可以按照导航进行配置。

1.3 三方库

编辑 flutter 项目的 pubspec.yaml 文件, 在里面按照规定的格式添加第三方库的名称、版本, 运行 flutter package get 就可以获得相应的库。本设计中采用的三方库有:

- date_format:
- web_socket_channel:
- groovin_material_icons: ^1.1.5
- dio:
- shared_preferences: ^0.4.1
- event_bus: ^1.0.1

2 LNMP 环境

2.1 搭建 Nginx 静态服务器

登录 Linux 服务器（此处使用 CentOS 7.0），使用 yum 命令安装 Nginx：`yum install nginx -y`

nginx 配置文件默认目录为 `/etc/nginx/conf.d/default.conf`，编辑文件设置监听端口。

修改完成后，启动 Nginx：`nginx`

将 Nginx 设置为开机自动启动：`chkconfig nginx on`

2.2 安装 MySQL 数据库

使用 yum 安装 MySQL：`yum install mysql-server -y`

安装完成后，启动 MySQL 服务：`service mysqld restart`

设置 MySQL 账户 root 密码：`/usr/bin/mysqladmin -u root password 'Password'`

将 MySQL 设置为开机自动启动：`chkconfig mysqld on`

2.3 搭建 PHP 环境

使用 yum 安装 PHP：`yum install php php-fpm php-mysql -y`

安装之后，启动 PHP-FPM 进程：`service php-fpm start`

把 PHP-FPM 也设置成开机自动启动：`chkconfig php-fpm on`。PHP-FPM 默认监听 9000 端口

之后需要配置 Nginx：在 `/etc/nginx/conf.d` 目录中新建一个名为 `php.conf` 的文件，并配置 Nginx 端口，修改配置完成后，重启 nginx 服务：`service nginx restart`