计算机网络 - 项目报告

Somnia1337

项目名：Echo Unity Archivist | 谐声收藏家

项目仓库

1 所选项目： Project 2. Email User Agent

> ⓘ Requirements
>
> - Users can download emails from their email box through the agent.
> - Users can send their emails through the agent.
> - The agent can access at least 2 real email servers.

2 开发环境与工具

- 框架：Flutter 3.22.0（前端）
- 语言：Dart 3.4.0（前端），Rust 1.78.0（后端）
- IDE：Visual Studio Code 1.89.1
  - 核心扩展：Flutter，Dart，rust-analyzer

3 设计与实现

3.1 前后端通信

前后端功能分离，前端只负责处理用户点击事件并向后端发送请求，后端只负责处理请求并响应。

前后端通信的信息格式在 Protocol Buffers(ProtoBuf) 规范下定义，并使用命令行应用 Rust in Flutter(Rinf) 为两端自动生成相关代码。

例如，`./messages/user.proto` 中定义有一个消息 `EmailMetadata`：

```
// [RINF:RUST-SIGNAL]
message EmailMetadata {
  string uid = 1;
```

```
  string from = 2;
  string to = 3;
  string subject = 4;
  string date = 5;
}
```

注释 `[RINF:RUST-SIGNAL]` 指示了这是一个来自 Rust（后端）的消息，它表示一封邮件的元信息。

当用户在前端的收件箱页点击“刷新”按钮时，前端将向后端发送一个请求，后端下载一定数量（25 封）邮件的元信息并返回给前端、展示在屏幕上。

3.2 前端

3.2.1 思路

在声明式 GUI 框架 Flutter 中，需要声明页面中的元素、对事件的响应以及触发页面状态的改变。

3.2.2 逻辑组织

前端（图形化用户界面）全部在 Flutter 框架下、使用 Dart 编写，结构如下：

```
./lib
├─ main.dart     -- 启动页 & 主页(界面骨架)
├─ compose.dart  -- 写邮件页
├─ inbox.dart    -- 收件箱页
└─ settings.dart -- 设置页
```

具体页面：

文件	页面
`main.dart`	`SplashPage` 启动页 `MainPage` 主页(界面骨架)
`compose.dart`	`ComposePage` 邮件编辑页 `SentEmailDetailPage` 已发送邮件详情页
`inbox.dart`	`InboxPage` 收件箱页 `MailboxPage` 邮箱页 `RecvEmailDetailPage` 接收到邮件详情页
`settings.dart`	`SettingsPage` 设置页

3.3 后端

3.3.1 思路

后端的主函数应异步运行，在循环中持续接收来自前端的消息、执行操作并返回结果。

3.3.1 逻辑组织

后端（代表用户与邮件服务器通信）全部使用 Rust 编写，结构如下：

```
./native/hub/src
├─ lib.rs  -- 入口
└─ user.rs -- 核心逻辑
```

3.3.2 依赖

后端依赖的核心第三方 crate：

```toml
tokio_with_wasm = "0.4.3" # 异步运行时
imap = "3.0.0-alpha.14"   # 与 IMAP 服务器通信
lettre = "0.11.7"         # 与 SMTP 服务器通信
```

3.3.3 主函数

```rust
pub async fn login_as_new_user() {
    loop {
        // 作为新用户登录
        actions_after_login();
    }
}

async fn actions_after_login() {
    // 具体操作
}
```

3.3.4 `User` 结构体

`User` 结构体表示一个用户实体，是后端的核心元素：

```rust
#[derive(Clone)]
struct User {
    smtp_domain: String,
    imap_domain: String,
    email_addr: Address,
    password: String,
}
```

对 `User` 实现有下列方法：

```rust
impl User {
    /// 构建用户实例
    fn build(...)

    /// 连接到 SMTP 服务器
    fn connect_smtp(&self)

    /// 连接到 IMAP 服务器
    fn connect_imap(&self)

    /// 发送新邮件
    fn send(&self, ...)

    /// 从 IMAP 服务器获取收件箱列表
    fn fetch_mailboxes(&self, ...)

    /// 获取指定收件箱中所有邮件的元信息
    fn fetch_metadatas(&self, ...)

    /// 获取指定邮件的正文及附件信息
    fn fetch_detail(&self, ...)
}
```

4 用户手册

演示用账户及授权码：`2581063732@qq.com` | `gpfjzcmovdkleadi`

4.1 启动软件

双击运行 `eua_ui_v0.5.6` 文件夹下的 `eua_ui.exe`。

4.2 启动页

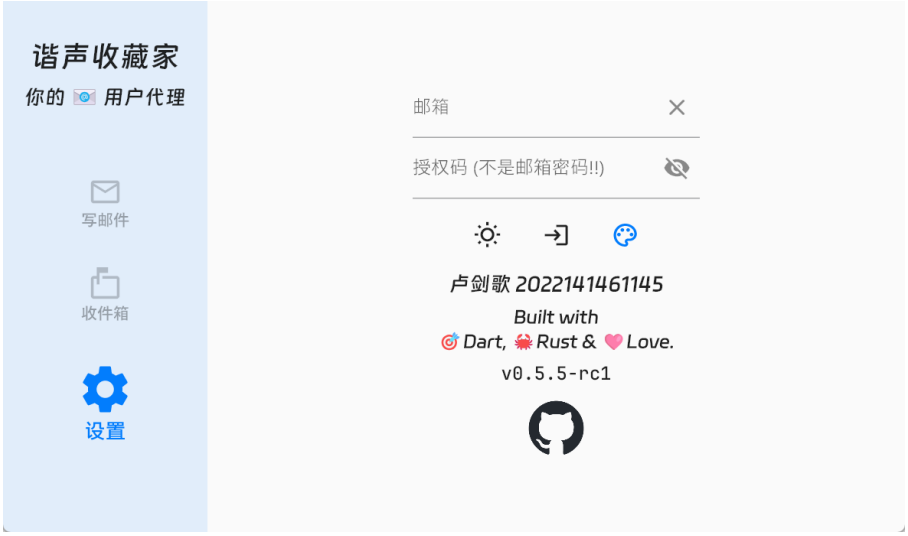
软件启动时显示一个封面：

等待 5 秒，随后将跳转到主页面。

4.3 主页面

主页面包含 GUI 的骨架元素。

窗口左侧有一个标题区，其下的导航栏导航至 3 个主要功能页面：

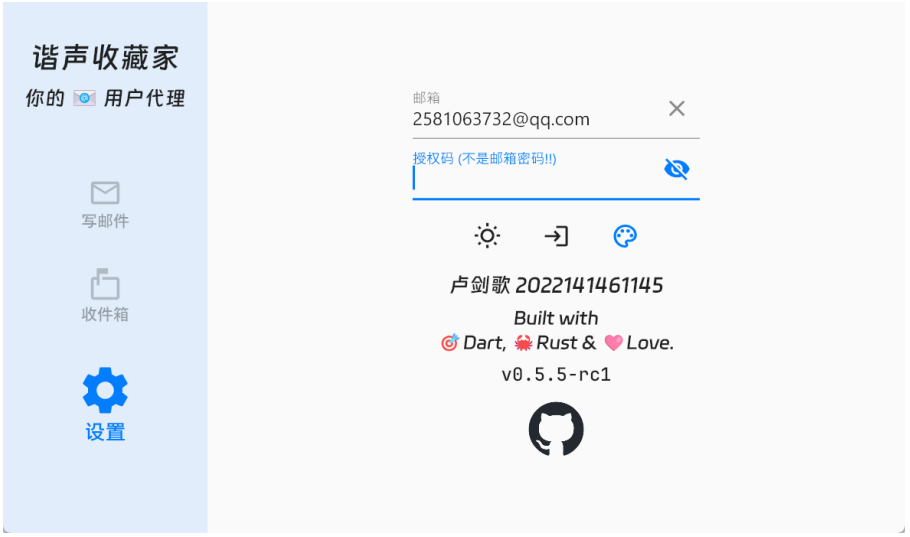


用户在登录前只能停留在设置页（其他页面的按钮被禁用）。

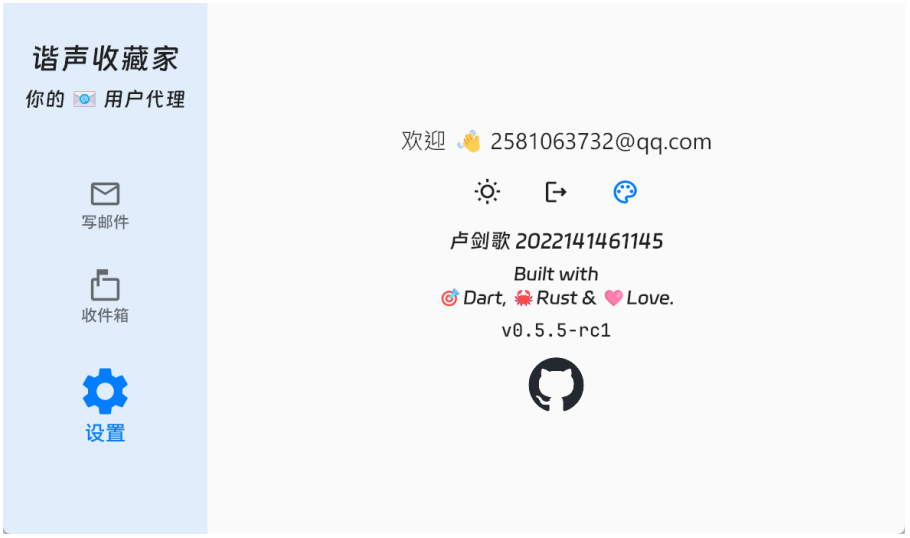
4.4 设置页

设置页根据是否已登录展现为 2 种形式：

- 登录前：

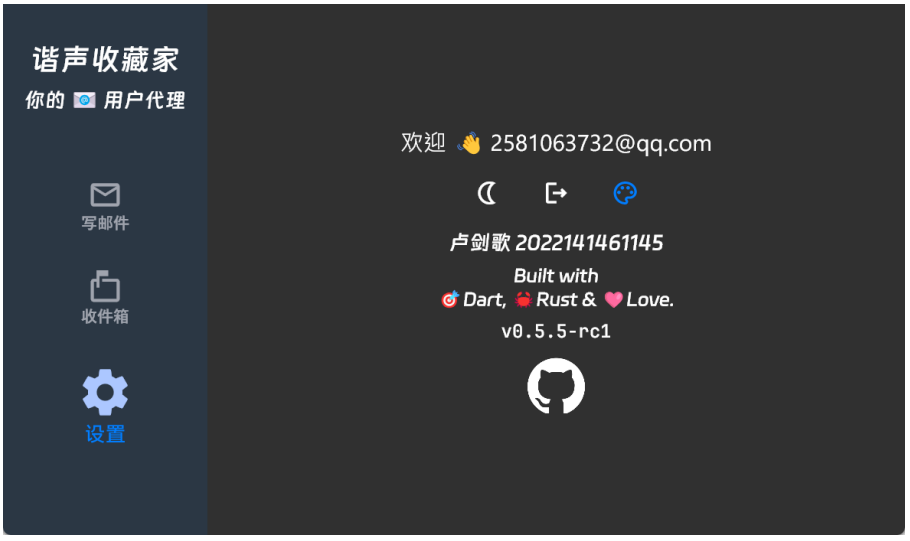


- 登录后：

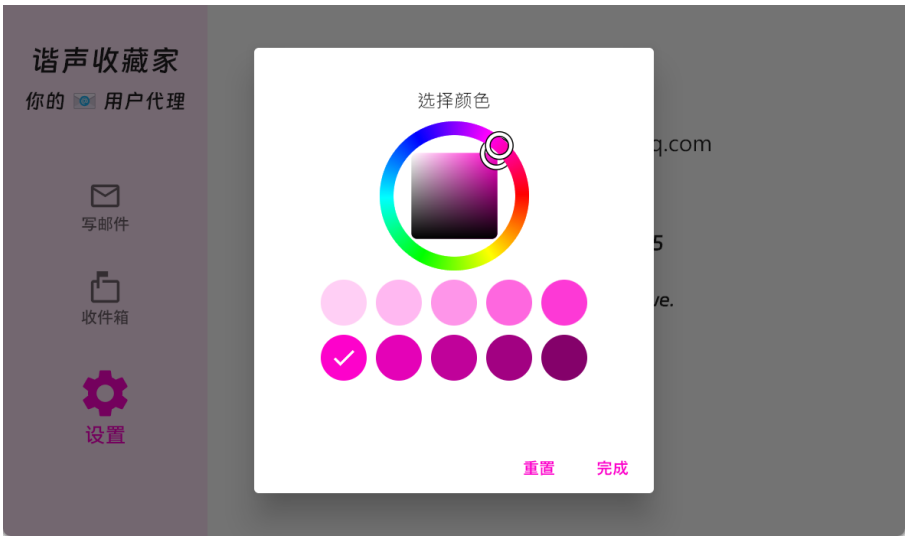


可选的设置项：

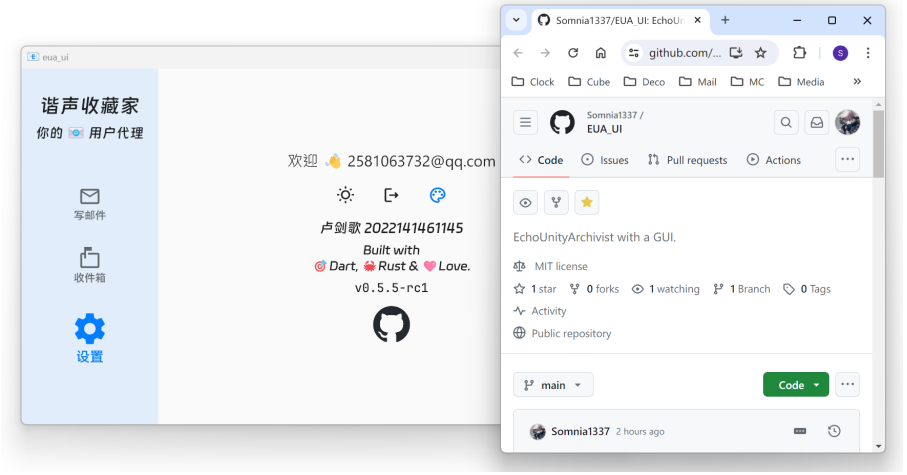
- 切换浅色模式/深色模式：



- 修改主题颜色：



单击页面下方的 GitHub 图标将跳转至项目仓库：



4.5 写邮件页

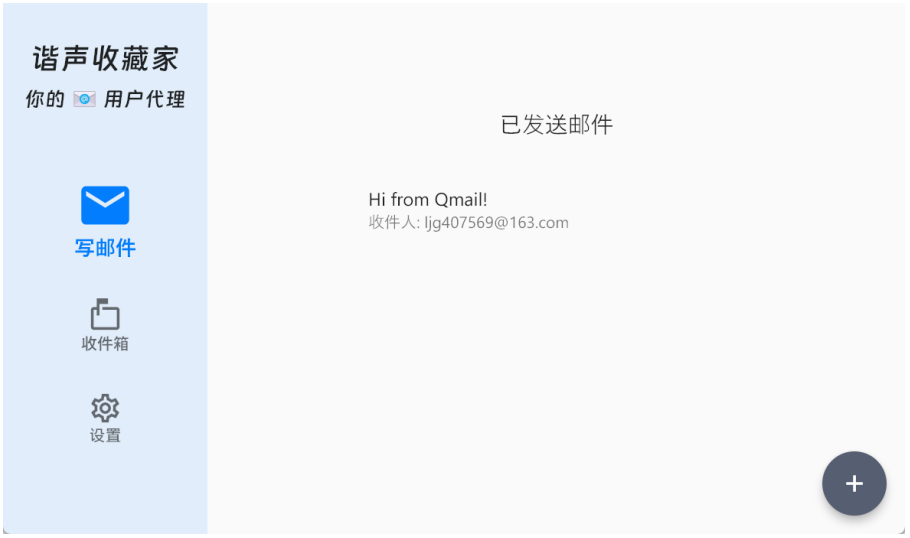
进入邮件编辑页时，需要单击右下角的“新邮件”按钮进入编辑区：

编辑区包含编辑新邮件所需的组件：

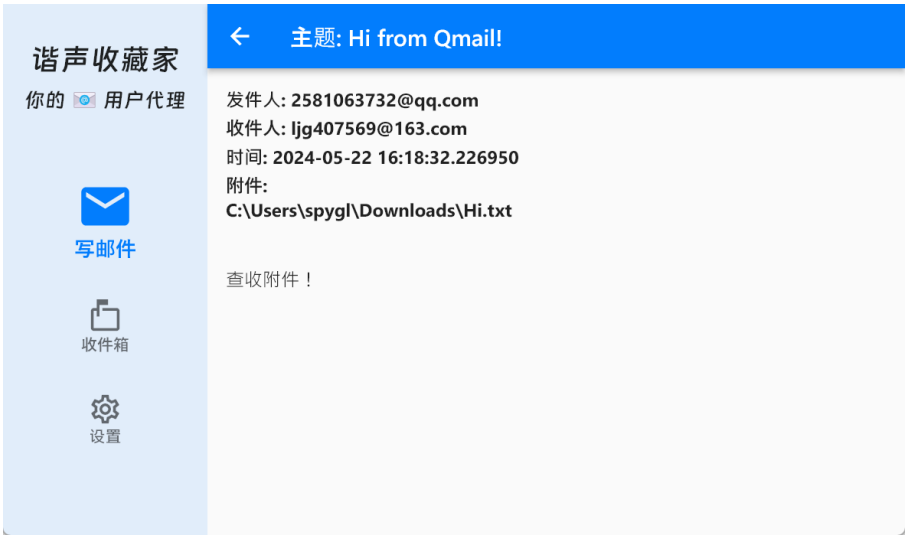
如果编辑了内容但未发送，退出编辑区时将提示是否保存：

编辑完成后，单击右下角的“发送”按钮即可发送：

发送后，软件自动切换到已发送邮件列表：

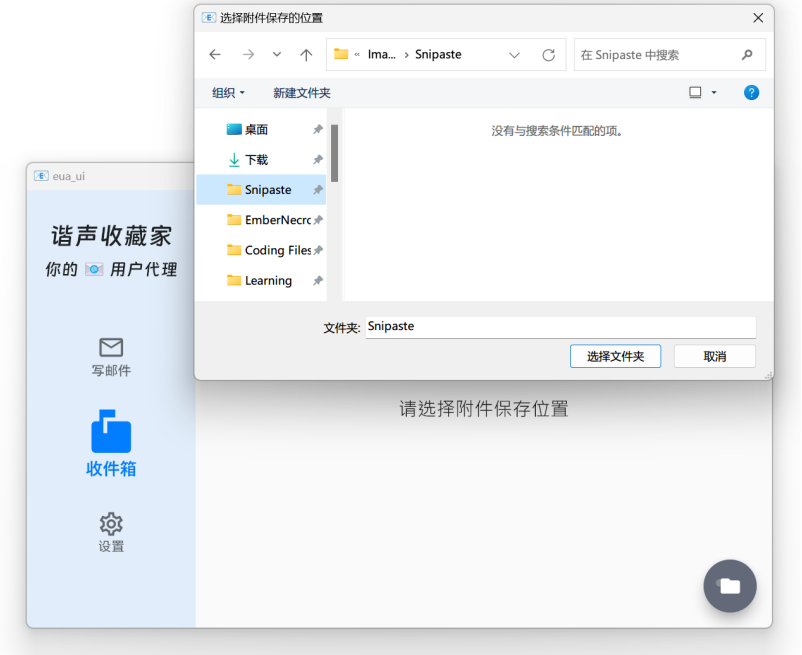


单击已发送的邮件，将展示其详细信息：

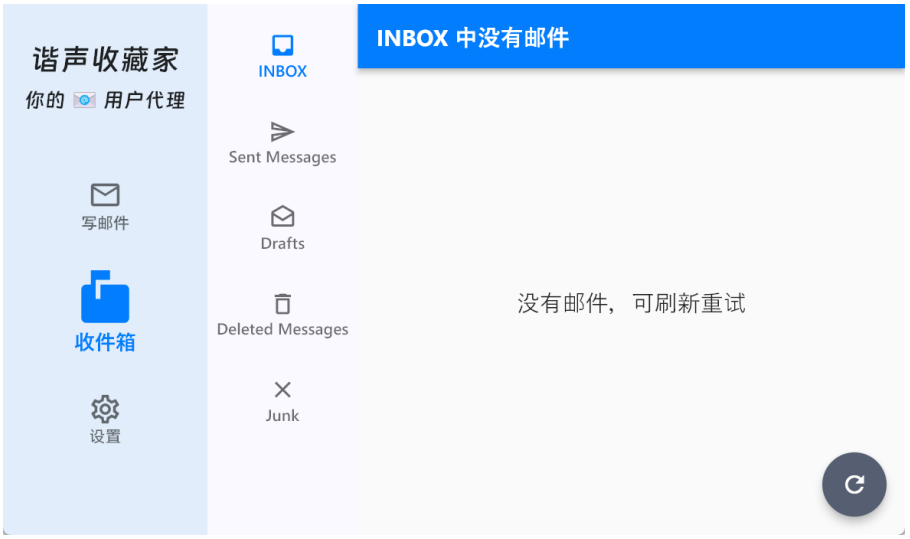


4.6 收件箱页

进入收件箱页时，需要先单击右下角的“选择位置”按钮指定附件保存位置：



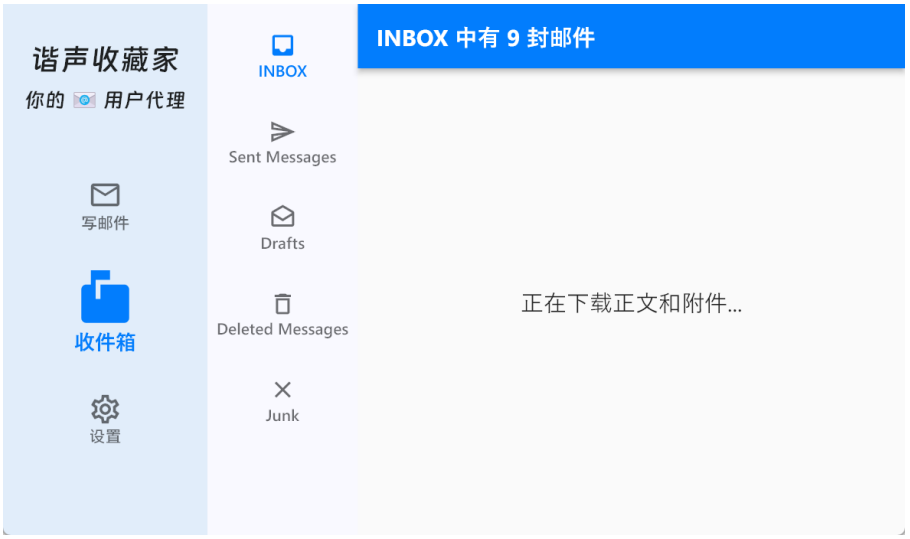
指定后，软件自动从服务器获取收件箱并展示在左侧边栏中，右侧展示邮箱内容：



单击右下角的“刷新”按钮将开始下载邮件元信息（每次最多 25 封），并实时更新已下载的数量：

下载完成后，将展示邮箱中的邮件列表：

单击其中一封邮件时，将下载邮件的正文及附件（附件保存在之前指定的位置）：



下载完成后，将展示其详细信息：