ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Diên

XÂY DỰNG ỨNG DỤNG UET MAIL TRÊN THIẾT BỊ DI ĐỘNG THÔNG MINH

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Truyền thông và mạng máy tính

Hà Nội – 2020

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Diên

XÂY DỰNG ỨNG DỤNG UET MAIL TRÊN THIẾT BỊ DI ĐỘNG THÔNG MINH

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Truyền thông và mạng máy tính

Cán bộ hướng dẫn: Ths. Đỗ Hoàng Kiên

Hà Nội – 2020

TÓM TẮT

Nhờ sự tiến hóa không ngừng của các thiết bị di động, từ những chiếc điện thoại di động đầu tiên nặng 4 đến 5 kg, cho đến những chiếc điện thoại bàn phím số có thể nghe gọi, nhắn tin... Hiện nay chúng ta đã sống trong một kỷ nguyên mới – kỷ nguyên của các thiết bị di động thông minh (smartphone). Chỉ với một chiếc smartphone nhỏ gọn trên tay, chúng ta đã có thể làm được hầu hết các công việc mà máy tính có thể làm được như: đọc, soạn thảo các văn bản Word, Excel,… ; trao đổi công việc thông qua Facebook Messenger, Viber, Zalo, Skype; tham gia họp trực tuyến thông qua Zoom, Microsoft Teams. Đặc biệt, nhờ sự tiến bộ của khoa học - kỹ thuật đã giúp cho chi phí sản xuất một chiếc smartphone ngày càng rẻ, ngày càng nhỏ gọn, thông minh. Chính những lý do đó đã khiến cho smartphone trở thành một phần không thể thiếu của mỗi người trong học tập, công việc hằng ngày.

Là một sinh viên UET-VNU (Đại học Công Nghệ - Đại học Quốc Gia Hà Nội), việc cập nhật những thông báo mới nhất của nhà trường thông qua kênh email đối với em cũng như các sinh viên khác là vô cùng cấp thiết và quan trọng. Do đó, em đã chọn đề tài "**Xây dựng ứng dụng uet mail trên thiết bị di động thông minh**" nhằm tạo ra một ứng dụng nhỏ gọn giúp sinh viên Đại học Công Nghệ có thể cập nhật email thông báo từ nhà trường tới sinh viên một cách nhanh chóng và tiện lợi. Ngoài ra việc xây dựng ứng dụng này cũng giúp em có thể tìm hiểu về các giao thức gửi, nhận mail qua mạng như SMTP, POP3, IMAP. Cùng với đó là việc thực hiện quy trình để thiết kế ra một ứng dụng mobile đơn giản dựa trên mô hình MVVM của Android.

Từ khóa: *Mobile, Android, UET Mail cho thiết bị di động thông minh*

LỜI CAM ĐOAN

Em xin cam đoan không sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ trong tài liệu tham khảo.

Hà Nội, ngày tháng năm

Sinh viên

Nguyễn Văn Diên

MỤC LỤC

LỜI MỞ ĐẦU 3
CHƯƠNG 1. TỔNG QUAN VỀ CÁC GIAO THỨC CỦA MAIL SERVER 4
1.1. Khái niệm về các giao thức của mail server 4
1.1.1. Khái niệm mail server 4
1.1.2. Các thành phần của mail server 4
1.1.3. Khái niệm mail client 6
1.1.4. Các giao thức bảo mật 6
1.2. Các giao thức của mail server 6
1.2.1. POP3 6
1.2.2. IMAP 7
1.2.3. SMTP 9
1.3. Các thao tác với email thông qua mail server 10
1.3.1. Cấu trúc cơ bản của email 10
1.3.2. Thiết lập kết nối tới mail server 10
1.3.3. Lấy email về 14
1.3.4. Gửi email 18
1.3.5. Trả lời email 19
1.3.6. Chuyển tiếp email 22
1.3.7. Di chuyển hoặc xóa email 22
1.4. Sử dụng hMail để tạo một mail server 23
CHƯƠNG 2. GSUITE VÀ GMAIL API 25
2.1. Giới thiệu về GSUITE 25
2.2. Gmail API và các thao tác với email thông qua Gmail API 25
2.2.2 Thiết lập kết nối tới Gmail API 26
2.2.3 Lấy email về 28
2.2.4 Gửi, trả lời và chuyển tiếp email 31

2.2.5. Di chuyển hoặc xóa email 32
CHƯƠNG 3. ANDROID VÀ MÔ HÌNH MVVM 33
2.3. Mở ứng dụng 35
2.4. Màn hình chính của ứng dụng 35
2.5. Thao tác với tài khoản email 37
2.6. Thao tác với email 39
KẾT LUẬN 41
TÀI LIỆU THAM KHẢO 42

LỜI MỞ ĐẦU

1. **Sơ lược về đề tài**
   - Tạo mail server thông qua hMail Server để có thể thử nghiệm quá trình nhận, gửi, quản lý email trong môi trường local.
   - Thực hiện kết nối với mail server để có thể nhận và gửi email thông qua các giao thức SMTP, POP3, IMAP.
   - Thực hiện nhận và gửi email của Gmail thông qua Gmail API.
   - Tạo ra ứng dụng android thông qua mô hình MVVM
2. **Mục tiêu đề tài**
   - Thiết kế một ứng dụng email dành riêng cho UET-VNU (Đại học Công Nghệ - Đại học Quốc Gia Hà Nội) với kích thước nhỏ gọn, tiện lợi cho việc nhận, gửi, quản lý email.
   - Nắm rõ được cơ chế hoạt động của các giao thức email như SMTP, POP3, IMAP.
   - Nắm được kiến thức về phát triển một ứng dụng android thông qua mô hình MVVM
3. **Đối tượng nghiên cứu**
   - Các giao thức giao thức mail: SMTP, POP3, IMAP.
   - Phần mềm hMail Server để tạo mail server.
   - GSUITE, cơ chế của GSUITE và cách kết nối tới Gmail API .
   - Ngôn ngữ lập trình Java và đặc biệt là Java Android để có thể thiết kế được một ứng dụng android.
   - Mô hình MVVM trong Android.
4. **Nội dung nghiên cứu**
   - Nghiên cứu giao thức email như SMTP, POP3, IMAP để có thể hiểu được bản chất và cách kết nối một mail client tới mail server.
   - Nghiên cứu về mail server, mail client và cách tạo mail server, cách cấu hình để kết nối từ mail client tới mail server được tạo.
   - Nghiên cứu về kiến trúc, mô hình thiết kế một ứng dụng Android và dựa vào đó để tạo ra ứng dụng UET Mail.

CHƯƠNG 1. TỔNG QUAN VỀ CÁC GIAO THỨC CỦA MAIL SERVER

1.1. Khái niệm về các giao thức của mail server

1.1.1. Khái niệm mail server

"*Mail Server là một hệ thống máy chủ được cấu hình riêng theo tên miền của tổ chức/doanh nghiệp dùng để gửi, nhận và quản lý thư điện tử.*"

Một mail server được tạo ra nhằm mục đích lưu trữ, gửi, nhận và quản lý toàn bộ email nội bộ của tổ chức/doanh nghiệp giúp cho toàn bộ email của tổ chức/doanh nghiệp đó được bảo mật, đảm bảo không được truy cập trái phép dữ liệu từ bên thứ ba cũng như tạo ra sự chuyên nghiệp cho tổ chức/doanh nghiệp đó thông qua tên miền riêng. Ví dụ: Đại học Quốc Gia Hà Nội có tên miền là vnu.edu.vn, mọi email của Đại học Quốc Gia Hà Nội đều kết thúc bằng @vnu.edu.vn.

Tuy có những ưu điểm như trên, mail server riêng của tổ chức/doanh nghiệp có một số nhược điểm sau:

- Tốn kém chi phí khi phải sử dụng server riêng để lưu trữ email thay vì sử dụng các dịch vụ email miễn phí như Gmail, Outlook, Yahoo mail,…
- Bảo mật thấp nếu như mail server không được bật các lớp bảo mật như SSL, TLS để tạo ra các kết nối an toàn.
- Phải tự xây dựng mail client riêng cho tổ chức/doanh nghiệp hoặc phải sử dụng mail client của bên thứ ba với cấu hình phức tạp

Hiện nay, ngoài việc tự tạo một mail server riêng thì còn có một dịch vụ rất hay của Google chính là bộ GSUITE. GSUITE giúp cho tổ chức/doanh nghiệp có thể có một tên miền riêng của họ, quản lý các tài khoản tại một server riêng mà tổ chức/doanh nghiệp tự thiết lập còn mọi dữ liệu về email sẽ được Google lưu trữ, chỉ khi người dùng kết nối thành công vào server của tổ chức/doanh nghiệp và được cấp quyền truy cập email của họ thì mới có thể truy cập, quản lý email. Đây là một bộ dịch vụ rất hữu ích, tạo ra sự chuyên nghiệp và đảm bảo tính bảo mật cao.

1.1.2. Các thành phần của mail server

Để có thể truyền, nhận và lưu trữ email thì một mail server sẽ phải bao gồm 2 thành phần chính sau:

- Incoming Mail Server (server thư đến): server này sử dụng giao thức IMAP hoặc POP3 để truyền mail từ server tới mail client, mail client cần cấu hình

đúng với cấu hình của server thư đến để có thể kết nối và tải email về. Nó được gọi là server thư đến vì đây chính là nơi mà mail client có thể lấy thư từ server về.

- Outgoing Mail Server (server thư đi): server này sử dụng giao thức SMTP để mail client có thể thiết lập kết nối và gửi email từ mail client tới mail server. Nó được gọi là server thư đi vì đây chính là nơi mà mail client có thể chuyển thư đi đến mail server và sau đó mail server sẽ chuyển tới một hoặc nhiều mail server chứa địa chỉ mail mà người dùng muốn gửi tới.

Tương ứng với mỗi server thư đến, server thư đi và thậm chí là tương ứng với mỗi giao thức (IMAP, POP3, SMTP) sẽ có các port (cổng) riêng để mail client kết nối vào. Tuy nhiên server thư đến và server thư đi có thể dùng chung một server (có chung địa chỉ IP/ hostname) nhằm tiết kiệm chi phí.

Để thực hiện một kết nối tới mail server thì ngoài việc cấu hình port và hostname như ở trên thì ta còn cần phải có account (tài khoản mail tương ứng cho mỗi người dùng tại mail server), password (mật khẩu) và email (chính là email của người dùng đó). Lưu ý là tên tài khoản có thể khác với email của người dùng.

Hình 1.1. Các thành phần của mail server

1.1.3. Khái niệm mail client

"Mail Client là một ứng dụng trên máy tính (desktop app), điện thoại di động (mobile app) hoặc ứng dụng web (web app) giúp người dùng có thể truy cập email, tiến hành quản lý email, duyệt email, gửi email."

Có rất nhiều mail client như Gmail, Outlook,… Để có thể kết nối với mail server thì cần phải cấu hình đúng với cấu hình mà tổ chức/doanh nghiệp đó cài đặt. Nếu muốn có các tính năng đặc thù cho mail client của tổ chức/doanh nghiệp thì cần phải tự xây dựng riêng một mail client cho tổ chức/doanh nghiệp đó.

1.1.4. Các giao thức bảo mật

Để có thể kết nối tới mail server một cách an toàn thì cần phải sử dụng giao thức bảo mật bổ sung. Hiện nay, có một số các giao thức bảo mật cho email như:

- SSL (Secure Sockets Layer) là tâng socket bảo mật được thiết kế để cung cấp truyền thông an toàn qua một mạng máy tính. Trong mô hình TCP/IP, SSL mã hóa dữ liệu của các kết nối mạng tại tầng ứng dụng (application). Còn trong mô hình OSI, SSL được khởi chạy ở tầng phiên (session) rồi hoạt động lên tầng trình diễn (presentation) nhờ đó mà dữ liệu truyền qua tầng giao vận (transport) được mã hóa và bảo mật.
- TLS (Transport Layer Security) là bảo mật tầng giao vận được thiết kế nhằm nâng cao và thay thế hoàn toàn SSL với các cơ chế bảo mật hơn
- Ngoài ra còn một số giao thức bảo mật cao hơn nữa được thiết kế nhằm tăng cường bảo mật cho giao thức SSL như: STARTTLS,…

1.2. Các giao thức của mail server

1.2.1. POP3

"*POP3 (Post Office Protocol version 3) là giao thức được sử dụng để kết nối tới mail server và tải email thông qua mail client.*"

POP3 là giao thức một chiều, có nghĩa là email được tải từ mail server xuống mail client và không thể xử lý các biên tập từ người dùng email. Việc sử dụng POP3 rất đơn giản, giao thức này chỉ thực hiện việc thiết lập kết nối và truyền mail tới mail client.

Khi sử dụng giao thức POP3 kết nối tới mail server để lấy email về mail client sẽ bao gồm các bước:

- Thiết lập kết nối đến server.
- Mail client tải toàn bộ email về và lưu trữ.
- Sau khi quá trình tải hoàn thành, mail client tiến hành ngắt kết nối và trong những lần sau email sẽ được đọc offline, nếu muốn cập nhật dữ liệu mới thì mail client cần quay trở lại bước ban đầu và kéo tiếp email về.

Khi kết nối tới mail server thông qua mail client bằng POP3 sẽ có tùy chọn để giữ mail trên server sau khi tải về. Nếu muốn truy cập mail trên nhiều thiết bị, người dùng nên chọn giữ lại bản copy trên server để các thiết bị khác có thể tải mail về được. Nếu không, mail sau khi tải về sẽ bị xóa sạch và không thể tải về từ các thiết bị khác.

Là một giao thức đơn giản, chỉ cần mail client yêu cầu kết nối và tải mail về lưu trữ và thao tác cục bộ tại client nên POP3 có một số ưu điểm như sau:

- Người dùng có thể truy cập ngay cả khi không có kết nối mạng do email được lưu cục bộ.
- Việc nhận và gửi mail vô cùng đơn giản.
- Server sẽ tiết kiệm được không gian lưu trữ.
- Cho phép lựa chọn việc có giữ lại bản sao mail trên mail server hay không.

Tuy rất đơn giản nhưng do có nhiều bất tiện ở giao thức này nên hiện nay POP3 không còn được sử dụng nhiều nữa mà thay vào đó là giao thức SMTP tuy phức tạp trong cơ chế kết nối nhưng rất tiện lợi cho người dùng.

Đối với giao thức POP3, có một số cổng mặc định để có thể kết nối như :

- Cổng 110 – Thường là cổng không mã hóa của POP3.
- Cổng 995 – Là cổng được mã hóa SSL/TLS của POP3.

1.2.2. IMAP

"*IMAP (Internet Message Access Protocol) là giao thức phức tạp hơn POP3 cho phép nhiều client cùng lúc kết nối tới một mail server. Email từ mail server sẽ được sao chép tới mail client và bản gốc của email vẫn sẽ được lưu trên mail server.*"

Giống như POP3, IMAP cũng được dùng để tải email từ mail server xuống mail client nhưng IMAP là giao thức 2 chiều, tức là ngoài việc tải email về IMAP còn cho phép xử lý các biên tập từ người dùng email chẳng hạn như : xóa mail, đánh dấu cờ cho mail (một email sẽ có một số cờ như Seen - đã đọc, Answered – đã trả lời,… giúp

người dùng có thể biết được trạng thái của mail đó), thậm chí là thêm một mail mới vào thư mục cụ thể.

Một điểm đặc biệt của IMAP là giao thức này có cơ chế để giúp cho người dùng có thể lưu trữ email vào các thư mục riêng (thư đến, thư spam, thùng rác,…) giúp cho việc quản lý email trở nên dễ dàng thay vì chỉ để mail ở một thư mục duy nhất là INBOX (thư đến) như ở POP3.

Khi sử dụng giao thức IMAP kết nối tới mail server để lấy và thao tác với email sẽ bao gồm các bước:

- Thiết lập kết nối đến server.
- Mail client lấy nội dung được yêu cầu về và lưu trữ.
- Sau khi quá trình lấy nội dung hoàn thành, mail client có thể thao tác để xử lý các biên tập người dùng như xóa các email đã được lấy, thêm email mới hoặc di chuyển email từ thư mục này sang thư mục khác.
- Mail client tiến hành ngắt kết nối.

Để thực hiện một số thao tác với email như xóa , trả lời, đánh dấu là đã đọc email ta cần phải nhớ một số system flag (cờ hệ thống) sau:

- \Seen (đã xem): cờ này cho ta biết rằng email đã được xem.
- \Answered (đã trả lời): cờ này cho ta biết rằng email (thư gửi đến) đã được trả lời.
- \Flagged (đã gắn cờ): email được gắn cờ này nhằm mục đích thể hiện đây là một email khẩn cấp/đặc biệt.
- \Deleted (đã xóa): khi email được gắn cờ này và truyền lên mail server thông qua IMAP thì nó sẽ được xóa ngay trên server.
- \Draft (nháp): cờ này thường được đánh dấu khi người dùng đang soạn thảo chưa xong email, mail client sẽ gắn cờ và lưu trữ tạm hoặc lưu trữ lên mail server để người dùng có thể soạn thảo tiếp vào lần sau.
- \Recent (gần đây): cờ này thể hiện email được gửi gần đây.

Tuy phức tạp nhưng giao thức IMAP giúp cho nhiều mail client từ các thiết bị khác nhau có thể truy cập, quản lý cùng một hộp thư đến. Do đó, IMAP có một số ưu điểm như sau:

- Mail server sẽ lưu trữ toàn bộ email của người dùng, việc lấy email từ mail server có thể thực hiện ở nhiều mail client.

- Việc lấy mail sẽ nhanh hơn nếu mail client chỉ lấy về các tiêu đề và lấy nội dung cụ thể của email theo yêu cầu từ người dùng.
- Mail client cũng có thể lấy toàn bộ tiêu đề và nội dung về lưu trữ cục bộ để có thể truy cập khi không có kết nối mạng.

Đối với giao thức IMAP, có một số cổng mặc định để có thể kết nối như :

- Cổng 143 – Thường là cổng không mã hóa của IMAP.
- Cổng 993 – Là cổng được mã hóa SSL/TLS của IMAP.

1.2.3. SMTP

"*SMTP (Simple Mail Transfer Protocol) là một giao thức chuẩn TCP/IP được dùng để truyền tải email trên mạng internet.*"

Giao thức SMTP thiết lập kênh kết nối giữa mail client và mail server, và thiết lập kênh liên lạc giữa mail server gửi và mail server nhận. Email sẽ được đẩy từ mail client lên mail server và từ mail server nó sẽ được server này gửi đến mail server nhận.

Khi sử dụng giao thức SMTP kết nối tới mail server để truyền email sẽ bao gồm các bước:

- Thiết lập kết nối đến server.
- Mail client thực hiện yêu cầu gửi email tới mail server. Mail server gửi sẽ nhận được yêu cầu và trước khi thực hiện truyền email tới mail server nhận, giao thức SMTP sẽ sao lưu toàn bộ email gửi đi ở SMTP-IN Queue (SMTP-IN Queue chính là nơi lưu trữ toàn bộ email được chuẩn bị gửi).
- Mail client tiến hành ngắt kết nối.

Ở mail server nhận, server sẽ tự động phân loại và sắp xếp email theo thứ tự ở local queue trước khi chuyển tới hộp thư của người nhận. Trong quá trình phân loại và sắp xếp email, mail server cũng sẽ thực hiện quét virus, kiểm tra spam để tăng cường bảo mật và giữ an toàn cho hệ thống.

Đối với giao thức SMTP, có một số cổng mặc định để có thể kết nối như :

- Cổng 25 – Thường là cổng không mã hóa của SMTP.
- Cổng 465 hoặc 587 – Là cổng được mã hóa SSL/TLS của SMTP.

1.3. Các thao tác với email thông qua mail server

1.3.1. Cấu trúc cơ bản của email

Để có thể thực hiện các thao tác với email, chúng ta cần hiểu cấu trúc cơ bản của chúng. Đối với một email chuẩn, sẽ có 2 thành phần chính như sau:

- Phần tiêu đề (headers) chứa các thông tin sau:
  - From (người gửi): thường chứa một địa chỉ email của người gửi.
  - To (người nhận): thường chứa một địa chỉ email của người nhận.
  - CC (carbon copy - người nhận): có thể chứa nhiều địa chỉ email của người nhận và họ cũng có thể xem được danh sách những người cũng nhận được email này.
  - BCC (blind carbon copy - người nhận ẩn): có thể chứa nhiều địa chỉ email của người nhận nhưng họ không thể xem được danh sách những người cũng nhận được email này.
- Phần nội dung (body) chứa các phần khác nhau, thường gọi là part.

1.3.2. Thiết lập kết nối tới mail server

Để thực hiện được các thao tác với email như nhận, gửi email thì bước đầu tiên cần làm là thiết lập kết nối tới mail server. Quá trình này được chia làm 2 phần:

- Thiết lập cấu hình kết nối tới incoming server:
  - Cấu hình tài khoản và mật khẩu kết nối tới incomming server.
  - Cấu hình hostname, port, loại kết nối (kết nối bảo mật thông qua SSL, TLS hay kết nối không bảo mật) tới incomming server và giao thức kết nối (đối với incomming server là IMAP hoặc POP3).
- Thiết lập cấu hình kết nối tới outgoing server:
  - Cấu hình tài khoản và mật khẩu kết nối tới outgoing server, đa số các mail server đều để tài khoản và mật khẩu outgoing server giống với tài khoản và mật khẩu của incomming server.
  - Cấu hình hostname, port, loại kết nối (kết nối bảo mật thông qua SSL, TLS hay kết nối không bảo mật) tới outgoing server và giao thức kết nối (đối với outgoing server là SMTP).

Ngoài ra, còn cần phải thiết lập email của người dùng vì email của người dùng có thể khác với tên tài khoản của họ. Ví dụ như email là abc@abc.com nhưng tài khoản có thể là abc chứ không phải abc@abc.com.

Sau khi thiết lập kết nối thành công tới mail server, một phiên kết nối (session) sẽ được tạo ra cho đến khi mail client thực hiện ngắt kết nối. Khi phiên kết nối được tạo ra, mail client có thể thực hiện các công việc như lấy toàn bộ email, xóa email, sửa email,… Chúng phụ thuộc vào các tính năng đã được thiết kế của mail client.

Dưới đây là code ví dụ về một class thực hiện chức năng thiết lập kết nối tới cả incomming và outgoing server với inbox, outbox có kiểu là UserModel chính là hai đối tượng chứa thông tin cấu hình người dùng của incomming và outgoing server:

```
package com.nc.uetmail.mail.session;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.MessagingException;

import javax.mail.NoSuchProviderException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Store;

import com.nc.uetmail.mail.database.models.UserModel;

import com.nc.uetmail.mail.database.models.UserModel.ConnectionType;

import com.nc.uetmail.mail.database.models.UserModel.MailProtocol;


public class MailSession {

   private static MailSession instance;


   private UserModel inbox;

   private UserModel outbox;

   private Session session;

   private Store store;


   private MailSession(final UserModel inbox, UserModel outbox) throws Exception {
```

```
        this.inbox = inbox;
        this.outbox = outbox;

        Authenticator auth = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(inbox.user, inbox.pass);
            }
        };

        Properties props = new Properties();
        String inProtocol = MailProtocol.valueOf(inbox.protocol).name;
        String ouProtocol = MailProtocol.valueOf(outbox.protocol).name;
        props.put("mail.store.protocol", inProtocol);

        props.put(String.format("mail.%s.auth", inProtocol), "true");
        props.put(String.format("mail.%s.host", inProtocol), inbox.hostname);
        props.put(String.format("mail.%s.port", inProtocol), String.valueOf(inbox.port));
        if ( ConnectionType.StartTLS.eq(inbox.type) ||
ConnectionType.SSL.eq(inbox.type)) {
            if (ConnectionType.StartTLS.eq(inbox.type)) {
                props.put(String.format("mail.%s.starttls.enable", inProtocol), "true");
            } else {
                props.put(String.format("mail.%s.ssl.enable", inProtocol), "true");
            }
```

```
        props.setProperty(String.format("mail.%s.socketFactory.class", inProtocol),
"javax.net.ssl.SSLSocketFactory");
        props.setProperty(String.format("mail.%s.socketFactory.fallback", inProtocol),
"false");
        props.setProperty(String.format("mail.%s.socketFactory.port", inProtocol),
String.valueOf(inbox.port));
    }

    props.put(String.format("mail.%s.auth", ouProtocol), "true");
    props.put(String.format("mail.%s.host", ouProtocol), outbox.hostname);
    props.put(String.format("mail.%s.port", ouProtocol),
String.valueOf(outbox.port));
    if (ConnectionType.StartTLS.eq(outbox.type) ||
ConnectionType.SSL.eq(outbox.type)) {
        if (ConnectionType.StartTLS.eq(outbox.type)) {
            props.put(String.format("mail.%s.starttls.enable", ouProtocol), "true");
        } else {
            props.put(String.format("mail.%s.ssl.enable", ouProtocol), "true");
        }
    }

    session = Session.getInstance(props, auth);
    try {
        store = session.getStore();
        store.connect();
    } catch (NoSuchProviderException e) {
        throw new Exception(e.getMessage());
    } catch (MessagingException e) {
```

```
            throw new Exception(e.getMessage());
        }
    }

    public static MailSession getInstance(UserModel inbox, UserModel outbox) throws
Exception {
        if (instance==null){
            instance = new MailSession(inbox, outbox);
        }
        return instance;
    }
}
```

1.3.3. Lấy email về

Đối với giao thức POP3, mail client chỉ có một sự lựa chọn duy nhất đó là lấy toàn bộ email về và lưu trữ chúng. Người dùng có thể đọc chúng mọi lúc, dù có kết nối mạng hay không vì toàn bộ email đã được tải về.

Còn với IMAP, mail client ngoài việc tải toàn bộ mail về như POP3 thì còn có thể lựa chọn tải về một phần các email hoặc chỉ cần tải tiêu đề (headers) để lấy về các thông tin quan trọng như : thư mới đến gần đây, tiêu đề thư, ngày thư được gửi, người gửi thư, người được gửi thư cùng (CC)…

Dưới đây là ví dụ về tiêu đề của một email trong hộp thư đến :

```
Return-Path: nc@ncmail.com
Received: from NCDOS (www.sublimetext.com [127.0.0.1])
  by localhost with ESMTP
  ; Sat, 28 Nov 2020 23:38:18 +0700
From: "Capta" <nc@ncmail.com>
To: "user" <user@ncmail.com>
Cc: <umail@ncmail.com>
Subject: Hello
Date: Sat, 28 Nov 2020 23:38:18 +0700
Message-ID:
<001801d6c5a4$e057fee0$a107fca0$@ncmail.com>
MIME-Version: 1.0
```

Đoạn code java dưới đây sẽ thực hiện việc lấy tiêu đề như trên và lưu trữ vào đối tượng MailModel (với session chính là phiên đã được thiết lập từ trước):

```
…
   public MailMessage(IMAPFolder imapFolder, Message message) throws Exception
{
      String attachFolder = MailAndroidUtils.ROOT_FOLDER + File.separator + new
Date().getTime();
      mailModel = new MailModel(
         -1, -1, message.getContentType(), imapFolder.getUID(message) + "",
         message.getSubject(), getAddressString(message, null),
         getAddressString(message, RecipientType.TO),
         getAddressString(message, RecipientType.CC),
         getAddressString(message, RecipientType.BCC),
         "", "", false, attachFolder,
         false, message.getFlags().hashCode(),
         message.getSentDate(), message.getReceivedDate(), true
      );
   }
   private String getAddressString(Message message, RecipientType type) throws
MessagingException {
      String address_str = "";
      Address[] addresses = type == null ? message.getFrom() :
message.getRecipients(type);
      if (addresses != null) {
         for (int i = 0; i < addresses.length; i++) {
            address_str += addresses[i].toString();
            if (i + 1 != addresses.length) address_str += ",";
         }
```

```
    }
    return address_str;
}
```

Thông qua việc chỉ tải về tiêu đề của IMAP, mail client có thể tải nhanh các thông tin quan trọng để người dùng có thể kịp thời kiểm tra và sau đó mới tải về nội dung của email khi người dùng mở mail trên mail client. Ngoài ra, mail client cũng có thể dùng cơ chế chỉ tải về tiêu đề để kiểm tra thư mới và thông báo cho người dùng để người dùng có thể cập nhật mọi lúc mà không cần phải mở mail client để kiểm tra.

Ngoài ra, IMAP còn có cơ chế phức tạp hơn đó là lấy email thuộc các thư mục đã được phân chia, giúp cho mail client có thể tải về một phần các email thay vì tải toàn bộ email ở một thư mục duy nhất là INBOX như POP3. Tuy nhiên, việc phân chia cấu trúc các thư mục không được thống nhất mà do mail client tự quy định. Ví dụ như: thư spam được một số phần mềm đặt tên thư mục là "spam" nhưng ở Microsoft Outlook đặt tên là Junk E-mail.Chính vì sự không nhất quán này nên người dùng cần phải sử dụng các phầm mềm mail client của cùng một bên thì sẽ tránh được sự bất tiện như: mail client phân loại thư mục nhầm nên khi chèn một email mới, nó sẽ được chuyển vào một thư mục khác.

Về body (phần nội dung cụ thể) của email, bao gồm rất nhiều phần phức tạp (được gọi là các part được liên kết với nhau, mỗi part đều có phần next part để trỏ từ part đầu tới các part tiếp theo) tùy thuộc vào nhu cầu của người gửi. Tuy nhiên, nội dung của một email thường gồm các loại part như sau:

- multipart/alternative: đa số phần đầu của nội dung email chứa phần này, nó thể hiện rằng đây là một email chứa nhiều phần (multipart).
- text/plain: mỗi email gửi đi ngoài địa chỉ người gửi, địa chỉ người nhận, tiêu đề thì đều cần có nội dung của mail và text/plain chính là nội dung mà người gửi soạn thảo tới người nhận dưới dạng text.
- text/html: chính là nội dung mà người gửi soạn thảo tới người nhận dưới dạng html.
- Ngoài ra là một số loại part khác thuộc dạng tệp tin như image/jpeg , image/png, application/pdf,… với nội dung tệp đều được mã hóa dạng base64 và thường xuất hiện khi người dùng đính kèm tệp hoặc chèn ảnh vào phần nội dung được soạn thảo.

Dưới đây là ví dụ về cấu trúc về body của một email đơn giản, chỉ được soạn thảo dạng văn bản thông thường bằng phần mềm Microsoft Outlook:

```
Content-Type: multipart/alternative;
  boundary="----=_NextPart_000_0019_01D6C5DF.8CB80F60"
X-Mailer: Microsoft Outlook 14.0
Thread-Index: AdbFpNpw+UU0Vht8TeK1Ju63wLn2LA==
Content-Language: vi

This is a multipart message in MIME format.

------=_NextPart_000_0019_01D6C5DF.8CB80F60
Content-Type: text/plain;
  charset="us-ascii"
Content-Transfer-Encoding: 7bit

Hello, how are you?


------=_NextPart_000_0019_01D6C5DF.8CB80F60
Content-Type: text/html;
  charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

<html xmlns:v=3D"urn:schemas-microsoft-com:vml" =
xmlns:o=3D"urn:schemas-microsoft-com:office:office" =
xmlns:w=3D"urn:schemas-microsoft-com:office:word" =
xmlns:m=3D"http://schemas.microsoft.com/office/2004/12
/omml" =
xmlns=3D"http://www.w3.org/TR/REC-html40"><head><META
=
HTTP-EQUIV=3D"Content-Type" CONTENT=3D"text/html; =
charset=3Dus-ascii"><meta name=3DGenerator
content=3D"Microsoft Word 14 =
(filtered medium)"><style><!--
/* Style Definitions */
p.MsoNormal, li.MsoNormal, div.MsoNormal
  {margin:0cm;
  margin-bottom:.0001pt;
  font-size:11.0pt;
  font-family:"Arial","sans-serif";
  mso-fareast-language:EN-US;}
a:link, span.MsoHyperlink
  {mso-style-priority:99;
  color:blue;
  text-decoration:underline;}
```

```
a:visited, span.MsoHyperlinkFollowed
  {mso-style-priority:99;
  color:purple;
  text-decoration:underline;}
span.EmailStyle17
  {mso-style-type:personal-compose;
  font-family:"Arial","sans-serif";
  color:windowtext;}
.MsoChpDefault
  {mso-style-type:export-only;
  font-family:"Arial","sans-serif";
  mso-fareast-language:EN-US;}
@page WordSection1
  {size:612.0pt 792.0pt;
  margin:72.0pt 72.0pt 72.0pt 72.0pt;}
div.WordSection1
  {page:WordSection1;}
--></style><!--[if gte mso 9]><xml>
<o:shapedefaults v:ext=3D"edit" spidmax=3D"1026" />
</xml><![endif]--><!--[if gte mso 9]><xml>
<o:shapelayout v:ext=3D"edit">
<o:idmap v:ext=3D"edit" data=3D"1" />
</o:shapelayout></xml><![endif]--></head><body
lang=3DVI link=3Dblue =
vlink=3Dpurple><div class=3DWordSection1><p
class=3DMsoNormal>Hello<span =
lang=3DEN-US>, how are
you?<o:p></o:p></span></p></div></body></html>
------=_NextPart_000_0019_01D6C5DF.8CB80F60—
```

1.3.4. Gửi email

Để thực hiện gửi email thì bắt buộc phải có các thông tin sau : email người gửi thư (đã được cấu hình từ trước) , email người nhận thư, tiêu đề thư, nội dung thư. Khi soạn email, người dùng phải nhập đầy đủ các thông tin này để có thể gửi email, đặc biệt là cần phập đúng email người nhận tránh trường hợp email được gửi nhầm địa chỉ.

Trong quá trình soạn email, nếu người dùng đóng trình soạn thảo thì mail client có thể lưu lại email đó dưới dạng bản nháp để người dùng có thể mở và soạn thảo tiếp. Hơn nữa, nếu có kết nối mạng thì mail client cũng có thể lưu trữ vào một thư mục nháp trên mail server để người dùng có thể soạn thảo tiếp trên các thiết bị khác thay vì chỉ lưu trữ cục bộ

Dựa vào phiên đã được thiết lập từ trước mà mail client có thể gửi thư tới người nhận nếu như email đã có đầy đủ các thông tin bắt buộc và người dùng thực hiện thao tác gửi.

Dưới đây là đoạn code java thực hiện việc tạo một email dựa vào các thông tin người dùng đã soạn thảo:

```
public static MimeMessage createMailMessage(
    Session session, MailModel newMail, HashMap<String, String> headers
) throws MessagingException {
    MimeMessage email = new MimeMessage(session);
    Flags flags = new Flags(Flags.Flag.SEEN);
    email.setSubject(newMail.mail_subject);
    email.setFrom(new InternetAddress(newMail.mail_from));
    email.setRecipients(RecipientType.TO,
MailMessage.toAddresses(newMail.mail_to));
    email.setRecipients(RecipientType.CC,
MailMessage.toAddresses(newMail.mail_cc));
    email.setRecipients(RecipientType.BCC,
MailMessage.toAddresses(newMail.mail_bcc));
    email.setSentDate(new Date());
    email.setText(newMail.mail_content_txt);
    email.setFlags(flags, true);
    return email;
}
```

1.3.5. Trả lời email

Việc trả lời email (Reply email) tương tự như việc gửi mail nhưng có một số điểm lưu ý về tiêu đề của email trả lời như sau :

- Phần tiêu đề email trả lời phải chứa trường “In-Reply-To” chính là trường “Message-ID” của email cần trả lời trước đó.

- Phần tiêu đề email trả lời còn phải chứa trường “References” chính là “Message-ID” của email cần trả lời trước đó nếu như email cần trả lời chưa có hoặc trống phần “References” hoặc là phần “References” và “Message-ID” của email cần trả lời trước đó nối tiếp với nhau bằng một khoảng trắng.
- Chủ đề (subject) của email trả lời thường bắt đầu bằng “Re: ”, theo sau là tiêu đề của email cần trả lời trước đó.

Dưới đây là ví dụ về một cuộc hội thoại bao gồm 3 email trao đổi giữa Join và Mary. Đầu tiên John gửi một email tới Mary, sau đó Mary trả lời cho Join bằng một email, cuối cùng Join trả lời lại Mary:

- Đầu tiên, John gửi một email tới Mary. Ta cần chú ý tới “Message-ID” của email này là <1234@local.machine.example>

```
----
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
----
```

- Tiếp theo, Mary gửi một email trả lời John. Ta thấy phần “In-Reply-To” và “References” chính là “Message-ID” của email trước đó (<1234@local.machine.example>); chủ đề của email là “Re: Saying Hello” chính là chủ đề của email được gửi trước đó thêm phần “Re: ” ở đầu.

```
----
From: Mary Smith <mary@example.net>
To: John Doe <jdoe@machine.example>
Reply-To: "Mary Smith: Personal Account"
<smith@home.example>
Subject: Re: Saying Hello
Date: Fri, 21 Nov 1997 10:01:10 -0600
Message-ID: <3456@example.net>
In-Reply-To: <1234@local.machine.example>
References: <1234@local.machine.example>

This is a reply to your hello.
----
```

- Cuối cùng, John lại gửi một email trả lời tới Mary. Ta thấy phần "In-Reply-To" chính là "Message-ID" của email trước đó (<3456@example.net>), còn "References" chính là "References" và "Message-ID" của email trước đó nối với nhau bởi một khoảng trắng (<1234@local.machine.example> <3456@example.net>).

```
----
To: "Mary Smith: Personal Account"
<smith@home.example>
From: John Doe <jdoe@machine.example>
Subject: Re: Saying Hello
Date: Fri, 21 Nov 1997 11:00:00 -0600
Message-ID: <abcd.1234@local.machine.tld>
In-Reply-To: <3456@example.net>
References: <1234@local.machine.example>
<3456@example.net>

This is a reply to your reply.
----
```

Ngoài ra, để trả lời tất cả những người được gửi (reply all), mail client chỉ cần lấy toàn bộ mail ở trường CC của mail cần trả lời cho vào trường CC của thư mới.

Dưới đây là đoạn code java thực hiện việc tạo một email để trả lời email đã nhận trước đó (với headers chứa các thông tin của email cần trả lời) :

…

```
public static MimeMessage createMailMessage(
    Session session, MailModel newMail, HashMap<String, String> headers
) throws MessagingException {
    MimeMessage email = new MimeMessage(session);
    Flags flags = new Flags(Flags.Flag.SEEN);
    email.setSubject(newMail.mail_subject);
    email.setFrom(new InternetAddress(newMail.mail_from));
    email.setRecipients(RecipientType.TO,
MailMessage.toAddresses(newMail.mail_to));
    email.setRecipients(RecipientType.CC,
MailMessage.toAddresses(newMail.mail_cc));
```

```
        email.setRecipients(RecipientType.BCC,
MailMessage.toAddresses(newMail.mail_bcc));
        email.setSentDate(new Date());
        email.setText(newMail.mail_content_txt);
        email.setSubject(headers.get("Subject"));
        flags.add(Flags.Flag.ANSWERED);
        email.setHeader("References", headers.get("References")+"
"+headers.get("Message-Id"));
        email.setHeader("In-Reply-To", headers.get("Message-Id"));
        email.setFlags(flags, true);
        return email;
    }
```

1.3.6. Chuyển tiếp email

Không phức tạp như việc trả lời email, quá trình chuyển tiếp email (Forward email) chỉ bao gồm việc lấy toàn bộ (copy) thông tin của email cần chuyển tiếp, sau đó bỏ phần BCC và thêm vào phần người nhận.

1.3.7. Di chuyển hoặc xóa email

Đối với việc di chuyển email từ thư mục này sang thư mục khác thì mail client cần thực hiện việc lấy thông tin email cần di chuyển về, lưu nó vào thư mục mới và sau đó xóa email ở thư mục cũ đi.

Để có thể xóa email thì ta chỉ cần lấy mail đó và gắn cờ \ Deleted. Tuy nhiên, đa số mail client hiện nay khi xóa email đều thực hiện việc chuyển chúng từ thư mục gốc tới một thư mục được coi là thùng rác để người dùng có thể khôi phục lại nếu lỡ tay xóa nhầm. Thư được lưu trữ trong thùng rác sẽ được xóa đi sau một khoảng thời gian tùy vào quy định của mail client.

Dưới đây là đoạn code java ví dụ về việc xóa hoàn toàn email khỏi server và client:

```
public void deleteMail(MailModel mailModel) throws Exception {
    if (mailModel == null) return;
    if (mailModel.mail_uid != null && !mailModel.mail_uid.isEmpty()) {
        FolderModel folderModel =
database.folderDao().getById(mailModel.folder_id);
        IMAPFolder folder = (IMAPFolder) store.getFolder(folderModel.fullName);
        folder.open(Folder.READ_WRITE);
        long mail_uid = Long.parseLong(mailModel.mail_uid);
        Message ms = folder.getMessageByUID(mail_uid);
        ms.setFlag(Flags.Flag.DELETED, true);
        folder.close();
    }
    database.mailDao().delete(mailModel);
}
```

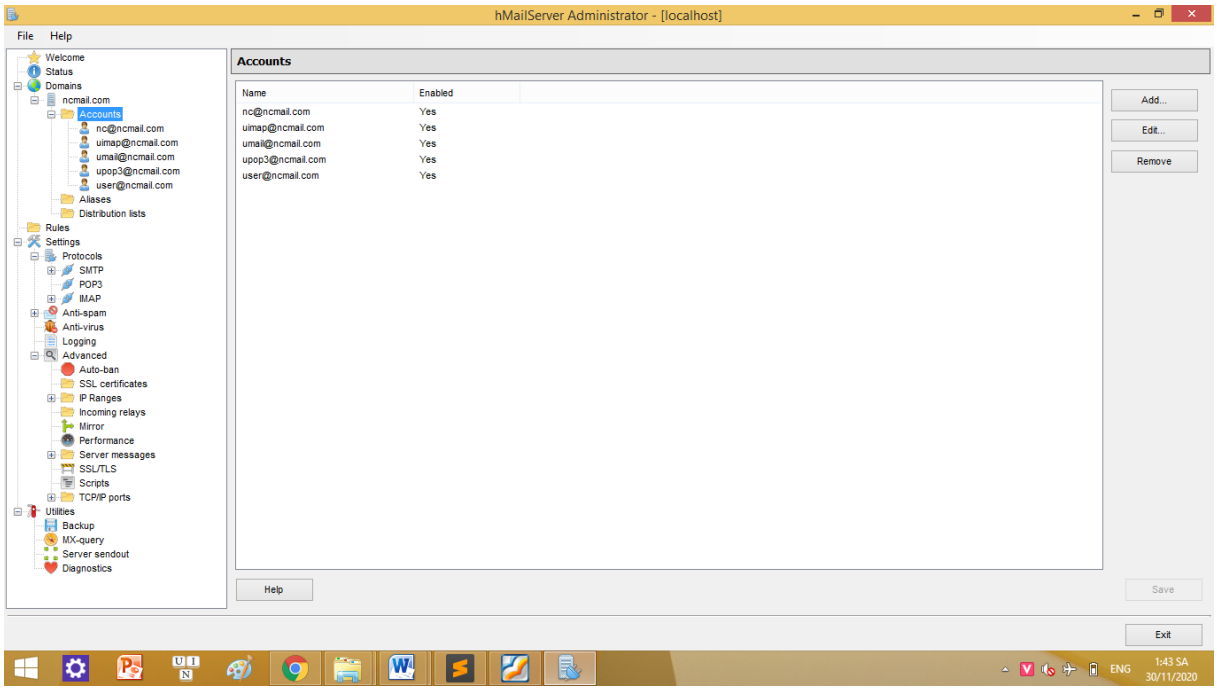
1.4. Sử dụng hMail để tạo một mail server

"*hMailServer là một phần mềm máy chủ email mã nguồn mở, miễn phí dành cho Microsoft Windows.*"

Để tạo một mail server phục vụ cho việc nghiên cứu, thậm chí là tại mail server thật thì chúng ta có thể lựa chọn phầm mềm hMail. Phần mềm này có những tính năng sau:

- Hỗ trợ các giao thức: POP3, SMTP, IMAP
- Hỗ trợ Virtual domains
- Hỗ trợ mã hóa SSL
- Hỗ trợ Anti-spam
- Hỗ trợ Anti-virus
- Có cơ chế backup dữ liệu
- …

Tuy nhiên có một điều đáng lưu ý là hMail server chỉ hỗ trợ MySQL 4 và MySQL 5. Đây là hai phiên bản MySQL khá cũ ở thời điểm hiện tại.



ABC

Sau khi tải, cài đặt và thực hiện kết nối hMail tới MySQL để có thể lưu trữ dữ liệu email thì chúng ta cần thiết lập để thêm một tên miền mới và thêm các tài khoản email vào. Ngoài ra, chúng ta cũng có thể thực hiện các thiết lập khác như thêm chứng chỉ SSL, cấu hình các cổng kết nối,…

CHƯƠNG 2. GSUITE VÀ GMAIL API

2.1. Giới thiệu về GSUITE

"GSUITE là một bộ ứng dụng điện toán đám mây và các công cụ phần mềm được cung cấp bởi Google trên cơ sở đăng ký thuê bao."

GSUITE bao gồm các ứng dụng web phổ biến của Google như Gmail, Google Drive, Google Hangouts, Google Calendar, và Google Docs. Mặc dù các ứng dụng này được cung cấp miễn phí cho người dùng thông thường, nhưng khi sử dụng GSUITE thì các ứng dụng sẽ được thêm rất nhiều các tính năng chuyên dùng như email tùy chỉnh theo tên miền của tổ chức/doanh nghiệp (@company.com), dung lượng lưu trữ email lớn.

Đây là một giải pháp điện toán đám mây vô cùng hữu ích cho các tổ chức/doanh nghiệp. GSUITE giúp cho mọi công việc trở nên nhanh chóng, năng suất cao nhờ việc lưu trữ mọi dữ liệu trên đám mây (các trung tâm dữ liệu bảo mật của google) thay vì lưu trữ cục bộ dữ liệu tại máy chủ của các tổ chức/doanh nghiệp.

Chính vì những ưu điểm như trên nên hiện nay chính tại trường Đại học Quốc Gia Hà Nội cũng đã thực hiện chuyển đổi sang các ứng dụng của GSUITE để thay thế các ứng dụng cũ, trong đó có một ứng dụng là UETMail. Trước đây, để phục vụ việc nhận và gửi mail thì toàn bộ các trường thành viên thuộc Đại học Quốc Gia Hà Nội đều phải tự thiết lập mail server riêng. Hiện nay, mọi việc trở nên đơn giản hơn với GSUITE vì ứng dụng Gmail của GSUITE có thể giúp chúng ta lưu trữ toàn bộ email trên các trung tâm dữ liệu của Google.

2.2. Gmail API và các thao tác với email thông qua Gmail API

2.2.1 Giới thiệu về Gmail API

"Gmail API là một RESTful API được dùng để truy cập vào Gmail mailboxes và thao tác với mail."

Để xây dựng một ứng dụng UETMail hiện nay thì chúng ta cần phải tìm hiểu về Gmail API rồi từ đó mới có thể thực hiện các chức năng thao tác với email.

Gmail API cung cấp cho chúng ta quyền truy cập một cách linh hoạt vào hộp thư của người dùng, với một số interface như Threads, Messages, Labels, Drafts, History, và. Settings. Chúng ta có thể sử dụng API để thực hiện các tính năng của Gmail như:

- Đọc thư từ Gmail.
- Gửi email.

- Sửa đổi các nhãn.
- Tìm kiếm tin nhắn
- Chuyển tiếp thư, trả lời thư,…

Các loại tài nguyên chính mà Gmail API cung cấp bao gồm:

- Messages: chứa thông tin và nội dung của email, messages là bất biến (chỉ có thể tạo hoặc xóa). Ngoài ra, không có thuộc tính nào trong phần tiêu đề có thể thay đổi ngoài các nhãn (labels).
- Labels chính là cách chính để phân loại và sắp xếp các thư và các chủ đề (Threads). Một nhãn có mối quan hệ nhiều-nhiều với thư và chủ đề: một chủ đề hoặc thư có thể có nhiều nhãn được áp dụng và một nhãn có thể được áp dụng cho nhiều thư hoặc chủ đề. Nhãn cũng có hai loại: system (nhãn hệ thống) và user (nhãn người dùng). Các nhãn hệ thống, chẳng hạn như INBOX, TRASH hoặc SPAM, được tạo nội bộ và không thể tạo, xóa hoặc sửa đổi. Còn đối với nhãn người dùng thì người dùng hoặc một ứng dụng có thể thêm, xóa hoặc sửa đổi.
- Drafts đại diện cho các thư chưa gửi. Bản thân các thư không thể sửa đổi sau khi được tạo, nhưng thư trong bản nháp có thể được sửa. Gửi bản nháp sẽ tự động xóa bản nháp và tạo một thư có nhãn SENT.
- History là một tập hợp các tin nhắn được sửa đổi gần đây theo thứ tự thời gian.
- Threads (chủ đề) là tập hợp các tin nhắn đại diện cho một cuộc hội thoại. Giống như thư, chủ đề cũng có thể có nhãn. Tuy nhiên, không giống như thư, không thể tạo chủ đề mà chỉ có thể xóa. Tuy nhiên, thư có thể được chèn vào một chủ đề.
- Settings (cài đặt) kiểm soát cách các tính năng của Gmail đối với người dùng. Các cài đặt có sẵn để truy cập POP và IMAP, chuyển tiếp email, bộ lọc, trả lời tự động trong kỳ nghỉ, bí danh gửi dưới dạng, chữ ký, đại biểu và ngôn ngữ.

2.2.2 Thiết lập kết nối tới Gmail API

Gmail API sử dụng OAuth 2.0 để xử lý xác thực và ủy quyền. Để kết nối thì mail client cần xác định một hoặc nhiều phạm vi (scopes) truy cập: chuỗi xác định các tài nguyên mà ứng dụng cần truy cập. Các phạm vi này được sử dụng cùng với một bộ tokens để đảm bảo quyền truy cập của người dùng vào tài nguyên.

Dưới đây là doạn mã thực hiện thiết lập kết nối tới Gmail api với một danh sách scopes cho trước:

```
List<String> SCOPES = Arrays.asList(new String[]{
    GmailScopes.MAIL_GOOGLE_COM,
    GmailScopes.GMAIL_ADDONS_CURRENT_ACTION_COMPOSE,
    GmailScopes.GMAIL_ADDONS_CURRENT_MESSAGE_ACTION,
    GmailScopes.GMAIL_ADDONS_CURRENT_MESSAGE_METADATA,
    GmailScopes.GMAIL_ADDONS_CURRENT_MESSAGE_READONLY,
    GmailScopes.GMAIL_MODIFY,
    GmailScopes.GMAIL_READONLY,
    GmailScopes.GMAIL_COMPOSE,
    GmailScopes.GMAIL_INSERT,
    GmailScopes.GMAIL_LABELS,
    GmailScopes.GMAIL_SEND,
    GmailScopes.GMAIL_SETTINGS_BASIC,
    GmailScopes.GMAIL_SETTINGS_SHARING,
});
HttpTransport HTTP_TRANSPORT = AndroidHttp.newCompatibleTransport();
GoogleAccountCredential credential = GoogleAccountCredential
    .usingOAuth2(context, SCOPES).setBackOff(new ExponentialBackOff());
credential.setSelectedAccountName(activeInbUser.user);
Gmail service = new Gmail.Builder(
    HTTP_TRANSPORT, JacksonFactory.getDefaultInstance(), credential
)
    .setApplicationName(context.getResources().getString(R.string.app_name))
    .build();
```

2.2.3 Lấy email về

Giống như việc lấy mail thông thường, sau khi thiết lập kết nối, mail client có thể chọn cách chỉ lấy phần tiêu đề hoặc lấy cả phần nội dung.

Dưới đây là đoạn mã java thực hiện việc bóc tách các thông tin của email được lấy về và lưu trữ vào cơ sở dữ liệu:

```
private void writePart(List<AttachmentModel> attachments, MailModel mailModel,
MessagePart p) throws IOException {
   if (p == null) return;
   writePart(attachments, mailModel, p, true);
}


private void writePart(List<AttachmentModel> attachments, MailModel mailModel,
MessagePart p,
                  boolean download)
   throws IOException {
   if (p == null) return;
   ContentType type;
   try {
      type = new ContentType(p.getMimeType());
   } catch (javax.mail.internet.ParseException e) {
      e.printStackTrace();
      return;
   }


   if (type.match("text/plain")) {
      mailModel.mail_content_txt =
StringUtils.newStringUtf8(p.getBody().decodeData());
   } else if (type.match("text/html")) {
```

```
        mailModel.mail_content_html =
StringUtils.newStringUtf8(p.getBody().decodeData());
    } else if (type.match("multipart/*")) {
        List<MessagePart> mp = p.getParts();
        for (int i = 0; i < mp.size(); i++) writePart(attachments, mailModel, mp.get(i));
    } else {
        String fileName = p.getFilename();
        if (fileName == null || fileName.isEmpty()) return;

        String path = mailModel.attachments_folder;
        if (!new File(path).isDirectory()) new File(path).mkdirs();
        if (!new File(path).exists()) throw new IOException("Can not make dir.");
        String filePath = path + File.separator + fileName;
        try {
            new File(filePath).getCanonicalPath();
        } catch (IOException e) {
            System.err.println(e.toString());
            fileName = fileName.replaceAll("[:\\\\/*?|<>]", "_");
            filePath = path + File.separator + fileName;
        }

        HashMap<String, String> hm = arrayToMap(p.getHeaders());
        String contentID = hm.get("Content-ID");
        contentID = contentID != null ? contentID.replaceAll("<|>", "") : null;
        boolean html_source = (mailModel.mail_content_html == null ||
            mailModel.mail_content_html.isEmpty() ||
            !mailModel.mail_content_html.contains(contentID)
```

```
        ) ? false : true;
        String attachId = p.getBody() != null ? p.getBody().getAttachmentId() : null;
        String uri = MailAndroidUtils.ROOT_URI + filePath;
        if (html_source && mailModel.mail_content_html != null &&
!"".equals(mailModel.mail_content_html))
            mailModel.mail_content_html =
                mailModel.mail_content_html.replaceAll("cid:" + contentID, fileName);
        attachments.add(new AttachmentModel(
            attachId, -1, p.getMimeType(), html_source, filePath,
            uri, (html_source || download), fileName,
            p.getBody().getSize(), hm.get("Content-Disposition"), contentID));
        if (hm.get("Content-Transfer-Encoding").equalsIgnoreCase("base64")
            && p.getBody().getSize() != 0 && (html_source || download)
        ) {
            if (attachId == null || attachId.isEmpty()) return;
            FileOutputStream fo = new FileOutputStream(filePath);
            MessagePartBody partBody = service.users().messages().attachments()
                .get(user, mailModel.mail_uid, attachId).execute();
            String data = partBody.getData();
            if ((null == data) || data.isEmpty()) return;
            byte[] decoder = Base64.decodeBase64(data);
            fo.write(decoder);
            fo.close();
        }
    }
}
```

```
private HashMap<String, String> arrayToMap(List<MessagePartHeader> headers) {
   HashMap<String, String> hm = new HashMap<>();
   if (headers == null) return hm;
   for (MessagePartHeader header : headers) {
      hm.put(header.getName(), header.getValue());
   }
   return hm;
}
```

2.2.4 Gửi, trả lời và chuyển tiếp email

Các bước thực hiện gửi, trả lời và chuyển tiếp email ở Gmail API không khác gì so với mail thường. Do đó, ta chỉ cần nắm rõ những trường bắt buộc của việc gửi email và bản chất của việc trả lời, chuyển tiếp email là có thể tạo ra các chức năng trên.

Dưới đây là một đoạn code java ví dụ về việc gửi email thông qua Gmail API:

```
private void send(
   FolderModel folderModel, MailModel newMail
) throws Exception {
   if (newMail == null) return;
   newMail.mail_flags_code = Flags.Flag.DRAFT.hashCode();
   newMail.id = 0;
   newMail.folder_id = folderModel.id;
   int newMailId = (int) database.mailDao().insert(newMail);
   newMail.id = newMailId;

   HashMap<String, String> hm = new HashMap<>();
   ByteArrayOutputStream buffer = new ByteArrayOutputStream();
   email.writeTo(buffer);
```

```
    String encodedEmail =
Base64.encodeBase64URLSafeString(buffer.toByteArray());
    Message sentMs = service.users().messages().send(user, new
Message().setRaw(encodedEmail))
        .execute();
    if (sentMs != null && sentMs.getId() != null) {
        newMail.mail_flags_code = email.getFlags().hashCode();
        newMail.mail_uid = sentMs.getId();
        database.mailDao().update(newMail);
    }
}
```

2.2.5. Di chuyển hoặc xóa email

Đối với Gmail API, sẽ không có khái niệm thư mục mà thay vào đó là khái niệm nhãn. Chúng ta có thể sửa đổi, cập nhật, thêm, xóa nhãn. Công việc này cũng tương đương với việc di chuyển một email, xóa hay sao chép email vào thư mục mới.

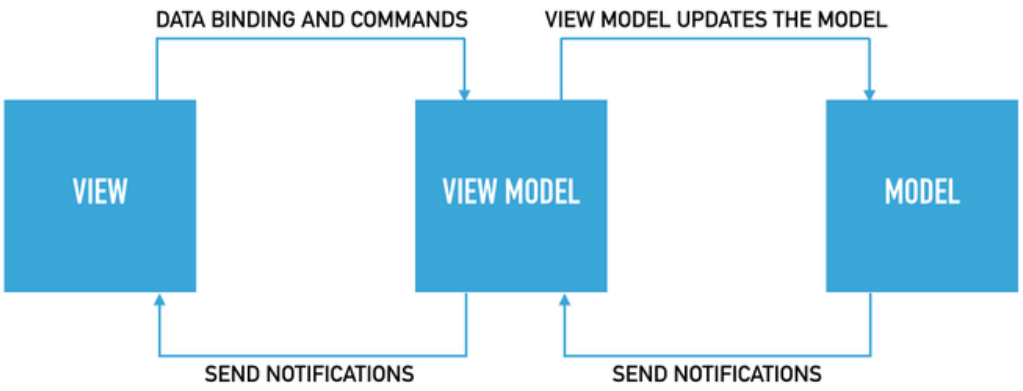
Ngoài ra, muốn xóa email thông qua Gmail API thì cần phải gọi tới API xóa email thay vì gắn cờ như ở email thông thường. Đặc biệt là việc cho email vào thùng rác cũng được Google cung cấp một API riêng để gọi tới.

Dưới đây là một hàm ví dụ dùng để xóa thư:

```
public void deleteMail(MailModel mailModel) throws Exception {
    if (mailModel == null || mailModel.mail_uid == null) return;
    database.mailDao().delete(mailModel);
    service.users().messages().delete(user, mailModel.mail_uid).execute();
}
```

CHƯƠNG 3. ANDROID VÀ MÔ HÌNH MVVM

- Mô hình MVVM
- Mô hình này khá giống với MVC (Model - View - Controller), sự khác biệt duy nhất là nằm ở cách xây dựng nên C (Controller) của MVC và VM (ViewModel) của MVVM.
- Model: Trong MVVM thì model sẽ thể hiện cho dữ liệu + trạng thái + các logic của đối tượng. Nó không có ràng buộc với View hoặc Controller vì vậy có thể được xử dụng lại dễ dàng
- View : Liên kết các biến quan sát và hành động bởi View Model. Quan hệ giữa View Model và View là 1-n, nghĩa là nhiều View có thể liên kết với 1 ViewModel
- ViewModel: Chứa các model và chuẩn bị các dữ liệu quan sát cho View. Nó cung cấp các móc để truyền dữ liệu từ View sang Model. Một điều cần phải ghi nhớ là ViewModel sẽ không ràng buộc vào View
-



2.3. Mở ứng dụng

Tại điện thoại, thực hiện nhấn vào ứng dụng có biểu tượng và tên như hình bên dưới để có thể mở UETMail:

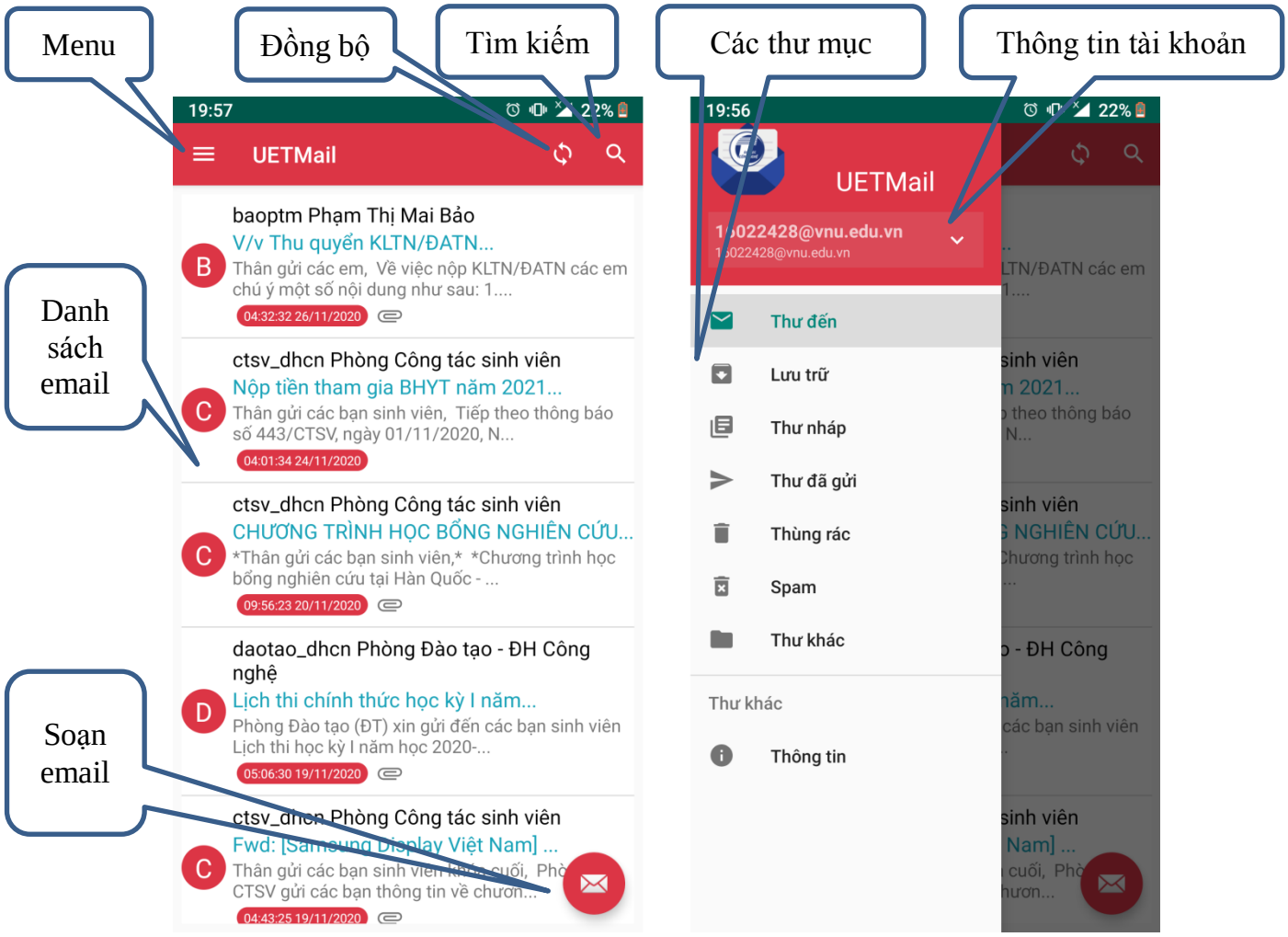
2.4. Màn hình chính của ứng dụng

Màn hình chính của ứng dụng UETMail bao gồm các thành phần sau:

- Nút mở "menu" để người dùng có thể lựa chọn tài khoản email và các thư mục email muốn xem.
- Nút "đồng bộ" để người dùng có thể đồng bộ thủ công toàn bộ email từ mail server về mail client nếu như việc đồng bộ tự động có vấn đề hoặc không hoạt động.
- Nút "tìm kiếm" giúp cho người dùng có thể tìm kiếm email trong mọi thư mục.
- Danh sách email là toàn bộ các email thuộc thư mục mà người dùng muốn xem, mỗi một email sẽ được hiển thị ngắn gọn các thông tin quan trọng như: người gửi, tiêu đề, tóm tắt nội dung, ngày gửi, email đã được xem hay chưa (hiển thị dạng icon), email có tệp đính kèm hay không (hiển thị dạng icon)…
- Nút "soạn email" là nút để người dùng hiện tại có thể soạn và gửi email.

Khi bấm vào "menu", thanh menu sẽ hiển thị tên ứng dụng, logo ứng dụng và các thành phần sau:

- Phần hiển thị thông tin tài khoản hiện tại dành cho mục đích quản lý nhiều tài khoản người dùng. Khi bấm vào phần này, người dùng có thể thực hiện các thao tác như thêm, sửa, xóa tài khoản email hoặc chọn tài khoản active (tài khoản hoạt động hiện tại).



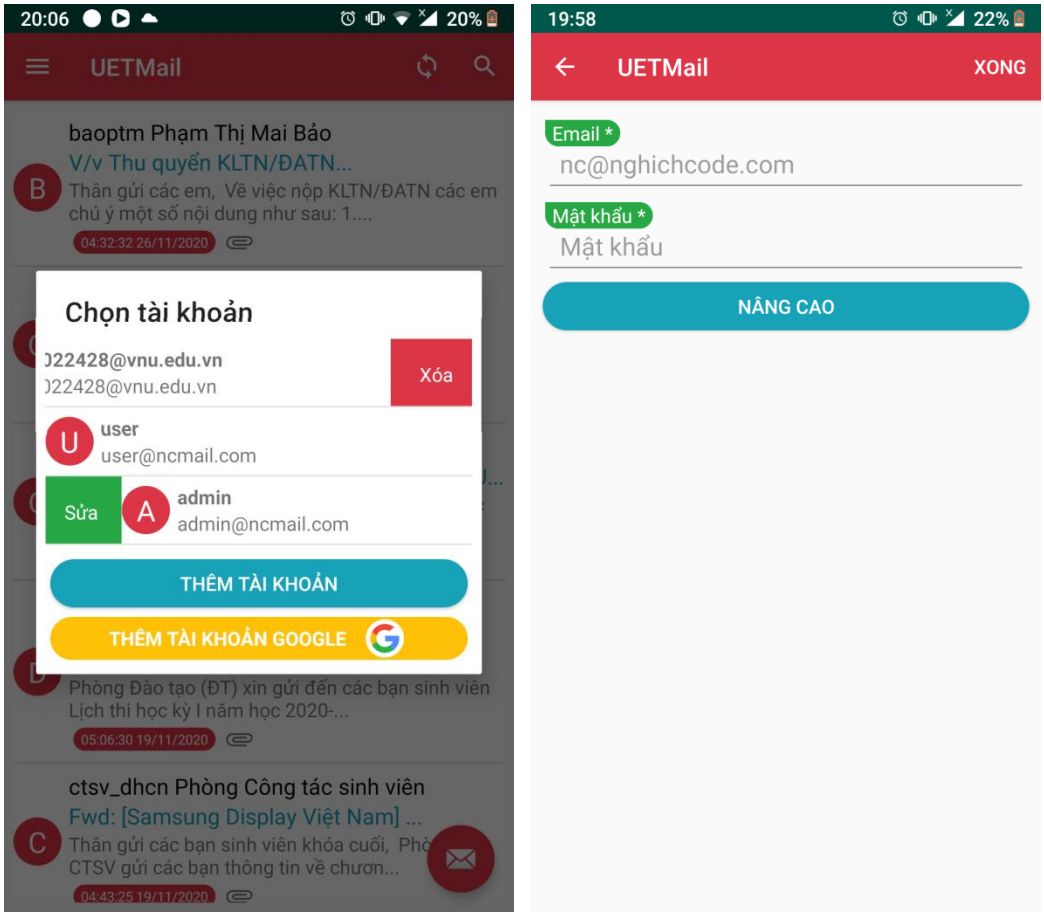
ABC

- Menu các thư mục quan trọng như:
    - Thư đến: Chứa các email được gửi đến.
    - Thư được lưu trữ: Thư mục riêng để lưu trữ email.
    - Thư nháp: Lưu trữ các bản nháp khi thư chưa soạn xong.
    - Thư đã gửi: Chứa các email đã gửi thành công.
    - Thùng rác: Các email trước khi bị xóa sẽ được lưu ở đây.
    - Thư khác: chứa các rmail và thư mục khác.
- Ngoài ra còn có nút hiển thị thông tin ứng dụng.

2.5. Thao tác với tài khoản email

Khi chọn vào phần tài khoản email, người dùng có thể thực hiện các thao tác sau:

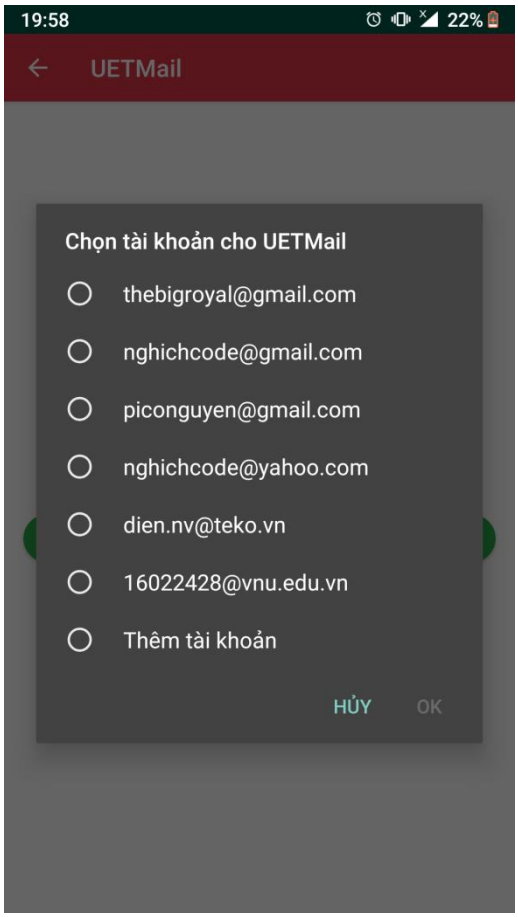
- Chọn tài khoản active (tài khoản hoạt động hiện tại). Khi một tài khoản được active, mọi phần mềm sẽ chỉ hiển thị toàn bộ email của tài khoản đó.
- Thêm , sửa, xóa tài khoản email.



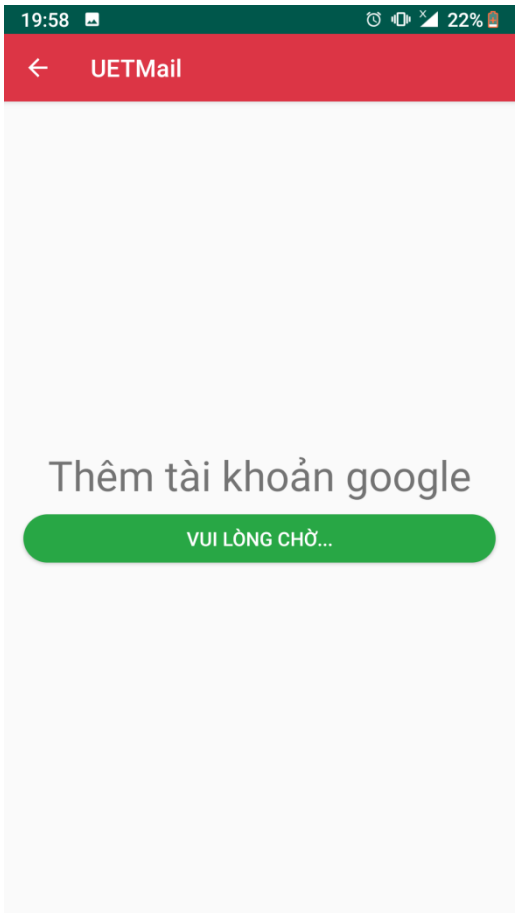
ABCEDE ABCD

Đối với việc thêm tài khoản email, sẽ có hai loại tài khoản sau:

- Tài khoản email thông thường: Người dùng sẽ phải cấu hình ít nhất hai trường bắt buộc là email và mật khẩu. Ngoài ra, người dùng còn có thể lựa chọn cấu hình nâng cao và thêm các thông tin như hostname, port, tên tài khoản, mật khẩu, loại kết nối của server thư đến và server thư đi sao cho phù hợp với cấu hình mà mail server quy định. Sau khi cấu hình xong, tài khoản sẽ được lưu trữ các thông tin trên và tự động tải email về.



ABC



ABC

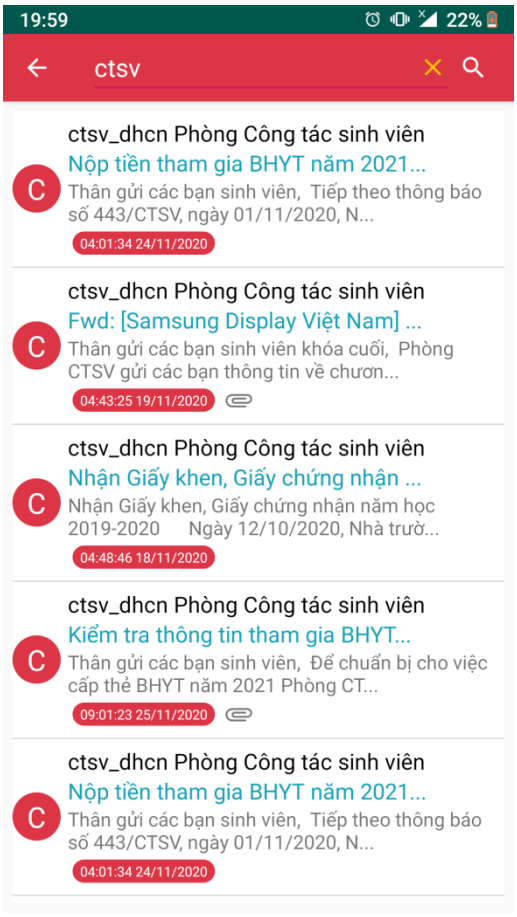
- Tài khoản email của google: Trước tiên, người dùng cần thêm thủ công tài khoản goooogle của mình tại phần cài đặt của điện thoại (phần này có các bước thực hiện tùy thuộc vào loại điện thoại đang dùng). Sau khi thêm thành công tài khoản google, người dùng chọn phần "Thêm tài khoản google" trên ứng dụng UETMail và sẽ được đưa đến màn hình liệt kê các tài khoản google đã được thêm trên điện thoại. Người dùng sẽ chọn một tài khoản trên màn hình để có thể thực hiện các thao tác với email. Sau khi chọn tài khoản google thành công, phần mềm UETMail sẽ tiến hành kiểm tra quyền truy cập email của tài khoản này. Nếu kiểm tra và kết nối thành công, phần mềm sẽ tự động tải email về và lưu trữ thông tin tài khoản phục vụ cho các kết nối sau.

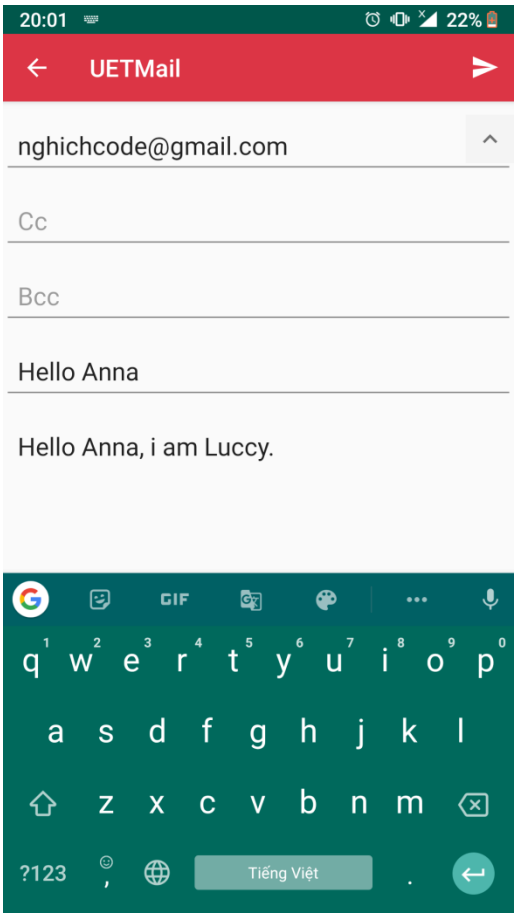
2.6. Thao tác với email

Đối với danh sách các email ở màn hình chính, người dùng có thể thực hiện một số thao tác với từng email như xóa hẳn email, cho email vào thùng rác, hoặc xem chi tiết nội dung email.

ABC ABC

Khi xem chi tiết email, người dùng cũng có thể thực hiện các thao tác như trả lời email, chuyển tiếp email đó.





ABC

Ngoài ra, người dùng có thể thực hiện tìm kiếm, soạn email bằng cách nhấn vào nút tìm kiếm hoặc nút soạn email ở màn hình chính. Khi soạn email, người dùng cần nhập ít nhất các thông tin như người nhận, tiêu đề, nội dung để có thể gửi được.

KẾT LUẬN

Thông qua việc tìm hiểu các thông tin về mail server và mail client cũng như việc cấu hình phần mềm hMail để tạo một mail server riêng, em đã có thể hiểu rõ được cơ chế tryền, nhận email cũng như cách mail server lưu trữ và các thao tác với email thông qua mail server. Đây là những kiến thức nền tảng vô cùng quý giá để em có thể thực hiện tìm hiểu về những thứ phức tạp hơn như việc tương tác với Gmail api. Nhờ vậy, em đã có thể hình dung và thực hiện các chức năng như tìm kiếm, thêm, sửa, xóa,… email cho ứng dụng UETMail của mình.

Ngoài ra, các kiến thức về lập trình android với mô hình MVVM cũng giúp ích rất nhiều cho em về việc thiết kế một ứng dụng trên thiết bị di động một cách hợp lý, cách tổ chức code mạch lạc phục vụ cho việc bảo trì ứng dụng cũng như làm cho ứng dụng hoạt động một cách nhanh chóng, chính xác.

Tuy nhiên, ứng dụng UETMail vẫn còn rất nhiều những tính năng thiếu sót, những vấn đề chưa được giải quyết. Những vấn đề này sẽ được em nghiên cứu và sửa chữa trong thời gian tới nhằm phục vụ tốt nhất nhu cầu sử dụng của sinh viên UET.

TÀI LIỆU THAM KHẢO

[1] https://quantrimang.com/phan-biet-pop-va-imap-89580

[2] https://wiki.matbao.net/mail-server-la-gi-moi-thong-tin-can-biet-khi-thue-mail-server/

[3] https://www.hostinger.vn/huong-dan/giai-thich-giao-thuc-pop3-smtp-imap-la-gi-va-port-cua-chung/

[4] https://tools.ietf.org/html/rfc3501

[5] https://www.ietf.org/rfc/rfc2822.txt

[6] https://vi.wikipedia.org/wiki/G_Suite

https://developers.google.com/gmail/api

https://viblo.asia/p/tim-hieu-mo-hinh-mvvm-trong-android-thong-qua-vi-du-phan-1-gioi-thieu-bWrZn67nZxw

MỤC LỤC

MAIL SERVER
- Send
- Forward
- Replly

GSUITE
GMAIL API
ANDROID MVVM
- ANDROID :
- - BROAD CAST

APP