Итакс.

Для начала в общих чертах опишу архитектуру проекта.

VombatiDB

Это моя встраиваемая графовая СУБД. Тебе с ней работать не придется, она здесь чисто для общего понимания структуры проекта. Она крутится под капотом у почтовика, реализуя масштабирование, персистентность, доступ к данным, индексацию, поиск, фильтрацию и сортировку (лол, кое что из этого еще не допилено, например сортировка). *Важное замечание: она написана на* `python 2.7`*. Да, он устарел, но так уж получилось. Само собой я буду переводить ее на* `python 3.8`*, но позже. Изза этой проблемы часть кода, который пишу я для проекта, пока что тоже на втором питоне.*

CapybaraMail

Это название всего проекта почтового клиента. Соответственно компоненты, которые входят в состав, именуются **CapybaraMail-***%component%*.

CapybaraMail-Store

Это модуль, реализующий работу с хранилищем почты, файлами (вложениями, аватарками и прочим, что накладно хранить в базе),

ярлыками, аккаунтами, а также в будущем любыми другими данными (например здесь же будет храниться база контактов или события календаря, если такой функционал будет добавлен когда нибуть). По сути - это заранее заданная схема базы данных и обертки над методами СУБД. Ну еще работа с файлами, да, единственная часть здесь не имеющая отношения к **VombatiDB**. Закончен не полностью, но основной функционал реализован. *Написан на `python 2.7` к сожалению.*

CapybaraMail-API

Этот модуль реализует более высокоуровневое апи над **CapybaraMail-Store**. Содержит базовую реализацию (которую предлагается импортировать в свое приложение и использовать как библиотеку) а также несколько (пока 1) сетевых версий - таже самая апи, но обернутая в тот или иной сетевой протокол и запускаемая во встроенном сервере (поверх TCP или *UDS). Например *REST* версия апи, это базовое апи обернутое в протокол `REST`, а *jsonrpc* соответственно будет обернуто в `jsonrpc`. Кроме того позже здесь же появится отдельная версия апи для *event-based* методов, например для уведомлений о новых письмах или об изменениях в ярлыках (ну например если почтовик открыт на нескольких компах и ты на одном из них создаешь новый ярлык). Закончены только базовые методы на чтение - тоесть для чтения ярлыков, аккаунта, почты и фильтрация. Но даже эти методы неоттестированы толком, а некоторые из них сейчас просто сигнатура и описание для документации, а внутри пустота. Но прописанные сигнатуры (название метода и аргументы которые он

принимает, их типы и тип возвращаемых данных) уже зафиксированы и меняться не будут. *Написан на* `python 2.7` *к сожалению.*

CapybaraMail-API-Server

Дополнительный модуль для запуска сетевых версий апи из **CapybaraMail-API**. Ничего примечательно, нужен чисто для удобства и на стороне удаленного сервера по сути будет запущен именно он, а он уже под капотом у себя поднимет все предыдущие пакеты.

CapybaraMail-WebClient-Backlog

А вот это уже клиентская часть проекта, которой я и предлагаю тебе заняться. На самом деле я планирую несколько клиентов разной степени упоротости и для различных платформ. Но это все планы, а начнем мы с именно этого - это пожалуй самая упоротая реализация - то как я вижу себе идеальный почтовый клиент для работы.

Технически это питоновый скрипт, который реализует основную логи и общается с удаленным **CapybaraMail-API-Server** по какому то сетевому протоколу. Этот же скрипт хостит SPA веб приложение на локалхосте и реализует промежуточное апи для него. Этот момент под вопросом, об этом в последней главе. Ну и еще он же будет показывать всплывающие уведомления, но это ваще потом ибо это тот еще гемор.

По сути - все письма сгруппированы в диалоги и отображаются ВСЕГДА в виде диалогов. Также есть 3+ папки:

- Входящие (сюда попадает вся новая почта, эта папка автоматически назначается всем непрочитанным диалогам).
- Задачи (она же бэклог, сюда помещаются письма, которые по сути являются заданиями, тасками и которые лежат там до тех пор, пока соответствующая им в реальности задача не будет выполнена).
- Заметки (сюда попадают письма, которые по сути памятки, например доступы к серверу или скан документа и т.д.)
- Иные, созданные пользователем, но вообще предполагается что этого хватит.

Если письмо не принадлежит ни одной из этих папок - оно находится в архиве, и доступ к нему можно получить только при помощи поиска или фильтров (о них ниже).

Перемещение между папками происходит вручную, или при помощи созданных пользователем специальных условий (Ну например для автоматических уведомлений от гитхаба о закрытии багрепорта, если существовало письмо). Также предполагается, что будут существовать специальные ярлыки, назначаемые автоматически при “архивировании” диалога из папки (кроме входящих канешно). Они нужны для возможности искать по, например, всем письмам в архиве, которые ранее были “задачами”. Автоматическое назначение этих ярлыков будет выполняться на стороне клиента, тобеш реализовывать его придется тебе.

При этом важно понимать - на уровне апи и ниже папок не существует - есть только ярлыки. Ну или категории или тэги, если так удобнее. Любое письмо может быть помечено любым количеством тегов. У диалогов нет тегов, считается что у диалога совокупность тегов из всех его сообщений. Как же тогда реализовать папки из описания выше - это те же самые ярлыки с заранее заданными названиями, их названия хранятся в настройках аккаунта (там можно хранить любые данные) и при запуске веб-приложение, запросив и прочитав эти настройки, далее считает что именно эти ярлыки являются условными папками.

Диалог считается непрочитанным, если в нем есть хотя бы одно непрочитанное сообщение. Если диалог помечен любой папкой кроме Входящих, и в него приходит новое письмо - он начинает отображаться сразу и там и там.

В любых иных случаях не допускается попадание диалога более чем в одну папку. Тоесть любой диалог может одновременно находиться только в одной папке, **за исключением папки Входящие**, она может быть назначена как второй. Данное ограничение введено специально, и на уровне апи и ниже (в том числе на уровне хранилища) его не существует, поэтому разруливать его тебе придется на своей стороне. Хотя еще не поздно, и если ты скажеш что оно сильно усложнит тебе разработку - я чтонить придумаю, например добавлю его как опциональное в хранилище.

Идем дальше, помимо папок есть и тру-ярлыки. Они

отрисовываются на сообщениях и по ним можно фильтровать сообщения. Это все ярлыки, кроме тех что мы считаем папками.

Далее есть система фильтрации. Сейчас она умеет фильтровать по любой комбинации ярлыков, дат, почтовых адресов отправителя, получателя и статуса прочитанности. Например чтобы отобразить всю почту, которая ранее была задачами, но была “выполнена”, мы устанавливаем соответствующую комбинацию ярлыков.

Для выбора условий фильтрации будет специальное окно, пока о нем не будем, я его еще не рисовал даже. Однако важно заметить, что составленные условия поиска можно будет сохранить как готовый пресет и задать ему имя.

Такие пресеты будут называть пользовательскими фильтрами. Некоторые из них будут заранее пред-конфигурированы для каждого нового аккаунта, например для просмотра всех сообщений вообще, всех отправленных, задач которые были “выполнены”, или помеченных как спам. По сути с технической стороны наши “папки” являются точно такимиже захардкоженными пресетами, просто отображаться они будут в интерфейсе отдельно.

Ну и еще кое что - в списке диалогов будет группировка по датам последней активности (то есть по последнему письму по сути). При этом подгрузка диалогов за следующую дату при прокрутке будет происходить лениво - ты просто текущий пресет фильтрации но указываешь другую дату. Это сделано не только для удобства, но скорее для снижения нагрузки.

CapybaraMail-TUI-Simple

Простой консольный интерфейс. Я начну писать его, как только закончу базовое апи. В первую очередь он нужен для тестирования. Хотя я планирую реализовать в нем весь основной функционал почтовых клиентов.

Вопросы

Я не уверен насчет нескольких пунктов. Как описано выше, сейчас я предполагаю:

*Технически это питоновый скрипт, который реализует основную логи и общается с удаленным **CapybaraMail-API-Server** по какому то сетевому протоколу. Этот же скрипт хостит SPA веб приложение на локалхосте и **реализует промежуточное апи** для него.*

Сделано это для того, чтобы было удобнее вынести сюда часть логики (например ограничение на пренадлежность только к одной папке) и при этом иметь возможность использовать удобный тебе сетевой протокол для связи с SPA или наоборот изучить и заюзать какой то новый, с которым ты бы хотел ознакомиться (например заюзать реактивность поверх вебсокетов, сейчас это весьма ценный навык). То есть с моим сервером ты общаешся по одному протоколу, а со своим веб-приложением - по другому.

Но само собой это займет у тебя дополнительное время, да и пилить

это придется на питоне. Ты готов к этому? Ты согласен? Не кажется ли тебе это лишней работой? Мы вполне можем отказаться от этого, и скрипт будет нужен только для уведомлений когда нибуть ну и локального хостинга SPA (в комплекте с питоном уже идет простой веб-сервер, для этого вабще ничего писать не придется). А всю дополнительную логику можно реализовать на стороне SPA.

Второй мой вопрос частично вытекает из первого. Чтобы не распылять силы, я сейчас буду реализовывать только дин сетевой протокол для апи. Какой? Я бы взял `jsonrpc`, но это мое личное предпочтение (очень уж я не люблю рест, а все остальные сильно сложнее для использования). Кроме того для него у меня уже есть куча инструментов, и мне попросту удобнее было бы с ним работать. На самом деле я бы вабще взял свою кастомную расширенную реализацию этого протокола, но это плохой выбор в данном случае - стандартность тут важнее фич.

Но я готов запилить первым любой иной протокол, если тебе с ним будет удобнее работать или если ты посчитаешь что он лучше подойдет. Ну или если ты захочешь получить опыт работы с каким то иным протоколом. Кроме того если мы откажемся от идеи промежуточного апи на стороне твоего приложения - то выбор протокола точно за тобой, с каким тебе будет удобнее работать.