MESSENGER

1. Чем руководствоваться при разбиении модуля на подмодули, при определении зависимостей и взаимосвязей различных модулей?

SOLID – принципы построения архитектуры проекта (смотри папку SOLID)

1. Как оформлять код, чтобы разные проекты, разрабатываемые разными людьми, соответствовали одному стилю?

Google C++ style guide (смотри папку Style guide) + код ревью проектов друг друга.

1. Как решить какую-то задачу проектирования, если архитектура какого-то модуля определена, но непонятно как реализовывать?

Книга «Приемы ООП. Паттерны проектирования» (смотри папку Паттерны). Хотя бы просмотрите сами паттерны и какие задачи они решают без изучения их до конца, чтобы представлять, что можно с их помощью делать.

Какой примерно алгоритм построения архитектуры

1. Определяем задачи модуля – для крупных модулей «верхнего» уровня их много => делим их на подмодули. Определяем основные качества модуля – то свойство, которым этот модуль должен обладать в первую очередь (учитывая то, какие задачи он решает): например, интерфейс пользователя надо легко уметь менять, поэтому его хотелось бы сделать так, чтобы можно было менять отдельные виджеты и то, чем они пользуются, без необходимости исправлять кучу файлов в нескольких модулях; для модуля, реализующего шифрование или обмен данными первостепенным качеством будет быстродействие.
2. Определились с решаемыми задачами, разделили модуль на подмодули, если надо. Прикидываем, какие связи у этого модуля есть с другими модулями (типа если у тебя есть массив сообщений, то их надо откуда-то получать, то тебе точно нужно уметь обращаться к сетевым функциям, либо сетевые функции должны иметь доступ к тебе).
3. Определись с зависимостями. Прикидываем, сколько классов необходимо для решения поставленных задач, набрасываем примерно интерфейсы этих классов (т.е. то, как этот класс виден «снаружи»).
4. Реализуем классы.

П. 1-2 скорее всего должны соответствовать SOLID.

В п. 3 с архитектурой модуля более-менее понятно, используем паттерны для построения иерархии классов, решающих поставленные задачи.

П. 4 кодим, используем стайл гайд.

Начинаем с клиентской части.

Держим в памяти, что часть модулей клиентской программы можно будет использовать и для сервера, так что прикидываем, что и как потом можно будет для сервера поменять.

Я проебался и диаграмма в конце получилась(

UI

Пользовательский интерфейс.

Содержит отображаемые виджеты + подключается к Local Data для обновления отображаемых данных.

По сути нам не надо иметь доступ полностью к Local Data, а иметь какого-то посредника для формирования запросов, для уведомления об обновлении данных и для получения данных.

Должно быть устроено так, чтобы можно было легко редактировать UI по желанию.

Local Data

Тут хранятся локальные данные приложения, в том числе настройки приложения (постоянные данные), данные сессии (можно что-то попробовать хранить локально, но все равно надо уметь синхронизироваться с сервером / другим пользователем).

Настройки приложения будут дорабатываться по ходу дела.

Данные сессии можно представить в виде: данные текущего пользователя, других пользователей, к которым подключен; данные бесед; данные подключений (более низкий уровень по сравнению с остальными, то, что получаем снаружи и парсим).

Предоставляет интерфейсы работы с данными для UI.

Имеет доступ к Functionality для выполнения сервисов типа установить подключение, синхронизировать данные, отправить свои данные и т.д.

Должно работать быстро

Functionality

Сервисы, выполняемые приложением.

По большей части все что, связано с установкой соединения и приемом/передачей данных.

ДОЛЖНО работать быстро и корректно.

UI

Functionality

Local Data

Application